Universitat Oberta de Catalunya

uoc.edu

Desarrollo de una aplicación Android para la gestión de un huerto domótico

Xavier Ledesma Pons
Máster en desarrollo de aplicaciones para dispositivos móviles
Trabajo final de máster

Pau Dominkovics Coll
Carles Garrigues Olivella

05/06/2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo de una aplicación Android para la gestión de un huerto domótico*
Nombre del autor:	*Xavier Ledesma Pons*
Nombre del consultor/a:	*Pau Dominkovics Coll*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Máster en desarrollo de aplicaciones para dispositivos móviles*
Área del Trabajo Final:	*Trabajo final de máster*
Idioma del trabajo:	*Castellano*
Palabras clave	*Huerto, domótica, riego*

Resumen del Trabajo

La aplicación, permitirá a personas de entornos particulares y profesionales realizar una automatización de sus zonas de cultivo, permitiendo personalizar y configurar las rutinas de riego de estos. Para ello, mediante placas Raspberry Pi y sensores de bajo coste, la aplicación permitirá configurar todo el entorno fácilmente, para cualquier usuario sin experiencia previa.

En el mercado no existen soluciones similares, ya que todo lo que se ofrece son aplicaciones para gestionar equipos profesionales de alto coste.

La aplicación, permitirá añadir el número de placas de riego necesarias según la superficie que se quiera cubrir. En cada placa de control, se podrán añadir los sensores y actuadores que el hardware permita y se podrán configurar en el sistema mediante la aplicación.

Con todo esto, se pueden crear rutinas de riego en base a fechas concretas de inicio y fin o en base a un valor concreto de un sensor.

Además, se permite exportar la configuración creada en la aplicación en un fichero o importar una configurar previa.

Para poder utilizar la aplicación, se deberá disponer de un servidor central, basado en una base de datos MongoDB y un servidor ExpressJS.

La aplicación se adaptará a dispositivos móviles y tabletas y estará disponible para versione de Android igual o superiores a la 7.0.

Abstract

The application, will allow people from private and professional environments to automate their growing areas, allowing them to customize and configure their irrigation routines. To do this, using Rasberry Pi boards and low-cost sensors, it will allow to configure the entire environment easily for any user without experience.

There’re no similar solutions on the market, since all available applications are for managing high-cost professional equipment.

The application will allow adding the necessary number of irrigation boards according to the surface to be covered. On each control board, the sensors and actuators that the hardware allows can be configured in the system through the application.

With all of this, you can create irrigation routines based on specific start and end dates or based on a specific sensor value.

In addition, it’s allowed to export the configuration created in a file or to import a previous configuration.

For using the application, you must have a central server, based on a MongoDB database and an ExpressJS server.

The application will be adapted to mobile and tablet devices and will be available for Android versions equal or greater than 7.0.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El proyecto que se va a desarrollar en este trabajo consiste en la creación de una aplicación Android para controlar y configurar un huerto domótico. A parte de la aplicación de gestión, se van a diseñar todas las partes de control del sistema siendo estas, el servidor de comunicaciones y el software que se ejecutara en la plataforma hardware (en el caso de este trabajo se ha elegido la placa Raspberry Pi 3).

Con este proyecto, se pretende crear un sistema suficientemente robusto y escalable, capaz de simplificar ciertas tareas a muchas personas que se dedican al sector de la agricultura, ya sea como hobby o profesionalmente y que a menudo, no pueden dedicar el tiempo suficiente a planificar y ejecutar tareas de riego o obtención de datos del cultivo.

Actualmente el sector del IOT y la industria 4.0, aparecen en muchos titulares, pero a menudo, se suelen enfocar estos avances a tareas como la gestión domestica o la automatización de ciertos procesos industriales, olvidando sectores como el agrario que son vitales para nuestra economía, pero que también necesitan de estos avances tecnológicos.

Hoy en día, existen diversas soluciones para la automatización de riego y la obtención de datos, pero algunas de estas carecen de interfaces intuitivas y simples. Otras más avanzadas, tienen un coste que muchos usuarios no pueden asumir.

Si se buscan soluciones al problema para la plataforma elegida, veremos que existen cantidad de tutoriales caseros sobre como montar sistemas similares al de estos proyectos. Solo con una primera lectura, podremos observar que muchos de estos presentan dificultades técnicas que muchos usuarios no son capaces de ejecutar.

El sistema propuesto, va a proporcionar a todos estos usuarios un entorno sencillo y muy escalable para la personalización de control deseada. Bastará con adquirir una placa Raspberry Pi y el número de sensores o actuadores deseados, y gracias a la aplicación, se podrá integrar todo de manera rápida, obteniendo en esta toda la información y pudiendo configurar nuestras preferencias de automatización de riego.

1.2 Objetivos del Trabajo

El objetivo principal del proyecto consiste en desarrollar una aplicación Android (móviles y tabletas) para la gestión de un sistema de control de riego automático, basado en la plataforma Raspberry Pi.

1.2.1 Objetivos funcionales

- La aplicación, se centrará en gestionar los diferentes elementos de una instalación de riego automática, permitiendo configurar sus elementos y visualizando los datos que estos obtengan.

- La aplicación permitirá gestionar el registro de los usuarios en el sistema y gestionar los usuarios de este.

- La aplicación se orienta a perfiles particulares y a perfiles profesionales.

- En ambos casos, el usuario podrá añadir N placas Raspberry Pi a su sistema de control, permitiendo incrementar el alcance físico del sistema.

- Se podrá seleccionar que modelo de Raspberry Pi se va a instalar, permitiendo compatibilidad con diversos modelos de esta.

- Se podrá añadir N actuadores y M sensores a cada placa mediante la aplicación.

- Se podrá programar el riego a partir de diversos mecanismos de configuración que se podrán combinar entre ellos.

- Se podrá configurar el inicio y finalización de riego a partir de una hora de inicio concreta y a partir de un evento de sensor (por ejemplo, que la temperatura sobrepase un limite).

- Se podrán visualizar logs del sistema, visualizando errores o eventos que sucedan en este.

- Se visualizarán en un panel los últimos datos recogidos por los sensores y el estado de los actuadores.

- Todo esto debe realizarse de forma segura mediante protocolos seguros de comunicación.

- La aplicación será multi idioma.

1.2.2 Objetivos no funcionales

- La aplicación será compatible con dispositivos móviles y tabletas Android.
- Todo el sistema será open source.
- Se escogerá un SDK que permita la ejecución en el mayor numero de dispositivos, pero manteniendo la funcionalidad original.
- La aplicación requerirá internet para funcionar.
- La información se almacenará en un servidor mediante una API Rest y una base de datos no relacional.
- La comunicación entre las placas y el servidor se realizará utilizando una API Rest.
- Para que el sistema funcione, el usuario se tendrá que registrar a través de la aplicación y posteriormente añadir las credenciales a la placa que quiera añadir.
- Una vez configuradas las credenciales, el usuario tendrá que ejecutar un script Python en la placa que quiera añadir.
- Mediante este mecanismo, se desacopla el servidor de las placas, y se permite tener el servidor ejecutando en una maquina con buena conexión a internet y las placas ejecutándose como clientes del servidor, mediante el envío de los mínimos paquetes de datos.

1.3 Enfoque y método seguido

Para la realización de este proyecto, se han estudiado diversas soluciones existentes en el mercado actualmente. A nivel no profesional, existen diversos tutoriales en internet que explican como integrar sensores y actuadores de bajo coste en plataformas como Raspberry Pi o Arduino, creando sistemas estáticos y de difícil adaptación a las necesidades requeridas.

En el caso de nuestra solución, se ha partido de estos entornos estáticos y se ha propuesto flexibilizarlos, mediante la gestión de un servidor y un software de comunicaciones. Adicionalmente, para poder tener una experiencia de personalización total, la aplicación Android va a diseñarse desde cero, adaptándose a los diversos escenarios de cada usuario.

Con esta estrategia se quiere mantener la esencia de la utilización de periféricos de bajo coste, al tiempo que se dinamiza y escala su utilización.

Finalmente se va a obtener un producto completamente nuevo y de bajo coste, capaz de obtener las características de equipos de gestión de riego inteligente profesionales, pero mucho más económicos, fáciles de adquirir y personalizables e instalables por cualquier usuario.

1.4 Planificación del Trabajo

Para el correcto desarrollo del proyecto, se han desglosado los distintos entregables en tareas, planificadas acorde con las fechas de entrega de cada uno de los 4 bloques principales.

Inicialmente, se va a proceder a un primer entregable, donde se va a diseñar el plan de trabajo del proyecto. Dentro de este primer bloque, las tareas principales serán:

- Propuesta del proyecto: se pensará que se quiere desarrollar y como hacerlo.

- Planificación del proyecto: se creará un plan de trabajo y una distribución de tareas para llegar de forma satisfactoria a cada entregable.

- Objetivos y enfoque: se empezará a redactar la memoria definiendo los principales objetivos y enfoques del proyecto.

A continuación, se presentará a un segundo entregable, donde se va a realizar un diseño abstracto del sistema y se va a definir su contexto de uso. Para ello, las tareas principales serán:

- Análisis de usuarios: se estudiarán los potenciales usuarios de nuestra solución para acabar de definir el uso de la aplicación.

- Diseño conceptual: se definirán los flujos de interacción del sistema a partir del análisis de los usuarios.

- Prototipo del sistema: se realizará un prototipo de la aplicación a partir de los flujos de interacción anteriores.

- Definición de casos de uso: se definirán los potenciales casos de uso que harán los usuarios de la aplicación.

- Diseño de arquitectura: se va a diseñar la arquitectura técnica a seguir en el desarrollo posterior.

En el tercer entregable, se realizará la parte de desarrollo del sistema. Para ello, las tareas principales serán:

- Desarrollo del servidor y integración con base de datos: se desarrollará el servidor y se integrará con la base de datos.

- Software de control de la placa Raspberry Pi: se desarrollará el software de comunicación entre la placa y el servidor.

- Aplicación Android: se desarrollará la aplicación Android de control y personalización del sistema.

- Pruebas de integración de todo el sistema: se probará que todo el sistema funciona correctamente y que la interacción entre todos los elementos es correcta.

- Redacción de la memoria técnica: se completará la parte técnica de la memoria, documentando el proceso de desarrollo.

Por último, se finalizará el proyecto y se elaborará la presentación de este. Para ello, las tareas principales serán:

- Finalización de la memoria: se procederá a terminar la memoria, completando los puntos que queden pendientes.

- Elaboración del manual del sistema: se elaborará un manual para poder replicar el entorno.

- Elaboración del manual de la aplicación: se elaborará un manual de como utilizar la aplicación móvil.

- Presentación: se elaborará una presentación sobre el desarrollo del proyecto y mostrando el producto final.

Para cumplir con los hitos anteriores, se va a seguir la planificación temporal de la Figura 1 y Figura 2.

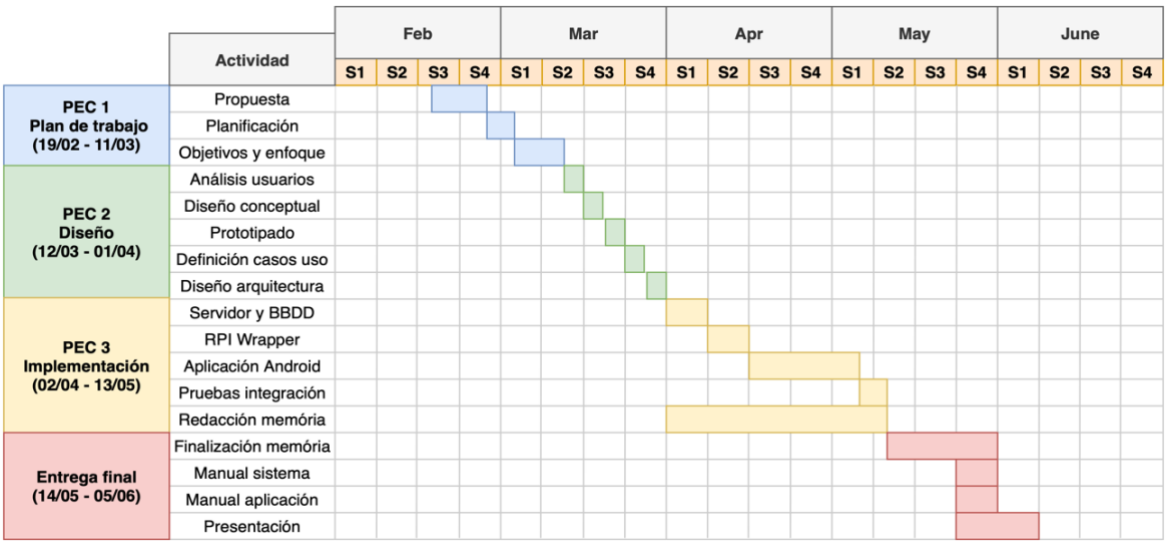


Figura 1: Diagrama Gantt del proyecto

PEC	Actividad	Horas laborables	Horas festivo	Horas totales
PEC 1 – Plan de trabajo (19/02 – 11/03)	Propuesta	4	0	4
	Planificación	2	4	6
	Objetivos y enfoque	4	8	12
PEC 2 – Diseño (12/03 – 01/04)	Análisis de usuarios	2	8	10
	Diseño conceptual	8	0	8
	Prototipado	2	8	10
	Definición casos de uso	6	0	6
	Diseño arquitectura	2	8	10
PEC3 – Implementación (02/04 – 13/05)	Servidor y BBDD	4	8	12
	RPI Wrapper	4	8	12
	Aplicación Android	10	16	26
	Pruebas integración	4	4	8

	Redacción memoria	16	0	16
Entrega final (14/05 – 05/06)	Finalización memoria	12	0	12
	Manual sistema	0	8	8
	Manual aplicación	0	8	8
	Presentación	8	8	16

TOTAL	88	96	184

Figura 2: Planificación temporal

Aproximadamente, se ha calculado una dedicación media de 6 horas durante la semana, es decir 3 horas 2 días a la semana, y de 6,6 horas durante el fin de semana, es decir 3,3 horas al día.

Posiblemente la carga de trabajo de los días laborables sea menor y se incrementen las horas los fines de semana.

1.5 Breve sumario de productos obtenidos

Al finalizar el proyecto, se obtendrá un sistema capaz de automatizar la gestión de un huerto o jardín de tamaño variable.

En este proyecto solo se trabajará con una placa de control con N dispositivos, pero se dejará el software preparado para poder añadir N placas con M dispositivos.

Como producto final, tendremos un sistema que nos permitirá a través de nuestro dispositivo móvil Android, configurar una infraestructura de riego automatizada con gestión horaria o con eventos de sensores del mismo sistema.

Adicionalmente, obtendremos el software que se ejecutará en las placas Raspberry Pi, permitiendo que estas se integren en la plataforma.

Además, el producto, ofrecerá la posibilidad de gestionar diferentes usuarios dentro de mismo entorno, facilitando la gestión en infraestructuras de tamaño considerable.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria del proyecto se estructurará de la siguiente manera:

- Introducción: en este apartado se expondrán los antecedentes al proyecto, como se va a planificar y cuales serán los hitos que se conseguirán.

- Diseño abstracto del sistema: en este apartado se analizarán posibles casos de uso del sistema y se justificarán las decisiones de diseño aplicadas.

- Diseño técnico del sistema: en este apartado se aportarán los detalles técnicos de cada parte del sistema y se justificará la elección de estos.

- Conclusiones: en este apartado se explicará como ha sido el desarrollo del proyecto y posibles ideas futuras de expansión.

- Glosario: en este apartado se incluirá la definición de acrónimos y términos técnicos utilizados en la memoria.

- Bibliografía: en este apartado se citarán las fuentes consultadas durante el desarrollo del proyecto.

- Anexos: en este apartado, se añadirán los manuales de uso y otra documentación relevante para la realización del proyecto.

2. Usuarios y contexto de uso

2.1 Tipos de usuario

La aplicación de este proyecto se orienta a usuarios con un perfil tecnológico bajo, únicamente se requiere de un perfil tecnológico de nivel medio para la primera puesta en marcha del sistema.

Una vez puesto en marcha gracias a la aplicación, toda la gestión del sistema la puede realizar el usuario que vaya a trabajar con él.

Según esto, podemos distinguir entre 2 usuarios distintos:

- Personal de cooperativas o del sector agrario, dedicados al cultivo y que administren plantaciones.

- Usuarios privados, que tienen una plantación y desean poder controlar el riego a distancia en función de sus preferencias.

2.2 Fichas de personas

Nombre	Luis
Edad	55
Profesión	Auxiliar administrativo

Luis trabaja como auxiliar administrativo en una empresa de utillajes para la automoción. Pasa gran parte del día al teléfono y cuando finaliza su jornada, gestiona un huerto que tiene en una población cercana a su domicilio.

Habitualmente, suele ir dos días a la semana para regar y luego un día el fin de semana para limpiar y recoger la cosecha.

Luis, busca en el mercado posibles soluciones para regar automáticamente y así ahorrarse los dos viajes de entresemana, pero todo lo que encuentra es muy caro y esta orientado al sector profesional. Además, son sistemas que requieren de instalaciones complejas y tiene que adquirir sensores muy caros. Todas las soluciones, requieren de conexiones a internet de banda ancha y él solo dispone de internet mediante tecnología 4G.

Personalmente, le gustaría poder gestionar de manera fácil y económica su huerto, pudiendo configurar el riego en función de la humedad y la temperatura, evitando malgastar agua si recientemente ha llovido o si la temperatura no es excesivamente alta.

Nombre	Carlos
Edad	60
Profesión	Propietario de una cooperativa

Carlos gestiona una cooperativa agrícola y trata de ayudar a agricultores a gestionar el tiempo y las labores para aumentar la producción. Debido a que sus productos son ecológicos y su producción muy limitada, deben cuidar el más mínimo detalle para que las perdidas sean mínimas.

Carlos esta buscando alguna manera de optimizar el proceso de riego de sus compañeros, para que puedan dedicar ese tiempo a otras labores. Como lleva muchos años en el sector, sabe que las inclemencias meteorológicas a veces pueden jugar malas pasadas y que, si a eso sumamos algunos errores humanos en el riego, el resultado puede ser devastador.

Para ello, ha pensado en ofrecer un servicio a los agricultores, de manera que ellos solo tengan que poner los dispositivos y cada uno pueda configurarlos según sus necesidades, evitando el sobrecoste de mantenimiento que pueda tener una instalación de tal magnitud.

De esta manera, le bastaría con dar una pequeña formación a cada uno y la cooperativa ya les proporcionaría el software necesario.

Tal como le han asesorado los informáticos, lo mejor sería que la cooperativa tuviera un servidor central, que este fuese seguro y que todos los agricultores lo usaran para gestionar sus cultivos.

Nombre	Teresa
Edad	35
Profesión	Abogada

Teresa trabaja como abogada en un bufete de abogados, situado en el centro de la ciudad donde vive. En casa tiene un pequeño jardín donde le gusta tener todo tipo de flores.

Últimamente, Teresa ha estado cubriendo diversos casos importantes y cuando ha llegado a casa, no ha tenido tiempo para regar las plantas que tiene.

Con el paso de los días, muchas de las flores que tenia se le han secado y ha tenido que replantar todo el jardín. Ella sabe que, si no busca una solución, en poco tiempo se le volverá a secar todo.

Ha estado consultando por internet y la única solución que ha encontrado han sido sistemas de riego gota a gota.

La idea no le acaba de gustar ya que, si llueve o hace humedad, no hace falta malgastar agua.

Además, como ha leído muchos libros de botánica, le gustaría estar informada de la temperatura y de la humedad de la tierra para saber cuando es el momento idóneo para plantar ciertos bulbos.

Como apenas tiene tiempo, le iría muy bien poder consultar todo con su dispositivo móvil y poder hacer sus configuraciones con él, aprovechando así los ratos libres que tiene en el trabajo y ayudándola a hacer las previsiones necesarias.

2.3 Listado de requisitos

Una vez se han analizado los potenciales usuarios de la aplicación, se presentan los requisitos funciones y no funcionales que esta debe de tener.

Como se trabaja con una metodología de trabajo iterativa, los requisitos pueden verse modificados en cualquier momento, ya sean para mejorarse o para añadir una funcionalidad necesaria.

Id	Requisito funcional
SG-RF1	Registro de usuario mediante nombre de usuario y contraseña.
SG-RF2	Configuración de IP y puerto del servidor central.
SG-RF3	Autenticación de usuario mediante nombre de usuario y contraseña.
SG-RF4	Ver listado de placas instaladas para el usuario en curso.
SG-RF5	Ver listado de sensores instalados para el usuario en curso.
SG-RF6	Ver listado de actuadores instalados para el usuario en curso.
SG-RF7	Ver listado de rutinas instaladas para el usuario en curso.
SG-RF8	Borrar una placa para el usuario en curso.
SG-RF9	Borrar un sensor para el usuario en curso.
SG-RF10	Borrar un actuador para el usuario en curso.
SG-RF11	Añadir una placa con un nombre personalizado y un controlador concreto.
SG-RF12	Añadir un sensor con un nombre personalizado, en una ubicación disponible y permitida por el controlador.

SG-RF13	Añadir un actuador con un nombre personalizado, en una ubicación disponible y permitida por el controlador.
SG-RF14	Añadir una rutina de riego a ejecutar por un actuador instalado, a partir de una fecha de inicio y una fecha final.
SG-RF15	Añadir una rutina de riego a ejecutar por un actuador instalado, a partir de un limite fijado en un sensor.
SG-RF16	Visualizar un log de eventos del sistema.
SG-RF17	Recibir notificaciones cuando se inicia una rutina de riego.
SG-RF18	Recibir notificaciones cuando se finaliza una rutina de riego.
SG-RF19	Exportar la configuración actual a un fichero.
SG-RF20	Importar una configuración anterior a partir de un fichero.
SG-RF21	Cerrar la sesión del usuario en curso.

Figura 3: Tabla de requisitos funcionales

Id	Requisito no funcional
SG-RNF1	La aplicación debe funcionar en dispositivos móviles con SDK mínima 24 (Android 7.0).
SG-RNF2	La aplicación debe funcionar en tabletas con SDK mínima 24 (Android 7.0).
SG-RNF3	La base de datos que se utilizará será MongoDB.
SG-RNF4	La API REST se realizará en NodeJS con el framework ExpressJS.
SG-RNF5	Las comunicaciones con la API se realizarán utilizando el protocolo HTTPS.
SG-RNF6	Las comunicaciones con la API REST utilizarán el formato JSON.
SG-RNF7	Las peticiones de cada usuario hacia la API deberán usar un token obtenido al inicio (JWT).

Figura 4: Tabla de requisitos no funcionales

3. Diseño conceptual

3.1 Escenarios de uso

En el siguiente apartado, se describen un conjunto de escenarios donde se hacen participes los usuarios descritos anteriormente.

Escenario 1
Luis con ayuda de su hijo, ha seguido el manual y ha configurado el servidor central en una Raspberry Pi que tenia por casa. Se ha instalado la aplicación, ha puesto los parámetros del servidor y se ha registrado en el sistema. Luego, ha ido al huerto y ha instalado otra Raspberry Pi con un par de sensores de humedad y temperatura que compro por internet. Además, compro una electroválvula que también se ha llevado. Con ayuda de su hijo, ha montado los componentes en los pines que tocaba y ha instalado un modulo 4G en la Raspberry para tener internet. A continuación, ha sacado su móvil y ha registrado la placa, los dispositivos y ha probado un par de rutinas de riego. Cuando ha visto que todo funcionaba se ha marchado a casa, y una vez allí, ha configurado el riego para que cuando uno de los sensores detecte que la humedad es inferior al 50% se inicie automáticamente el riego, forzando así a que la tierra siempre tenga un 50% de humedad. Además, como es verano, ha creado también una rutina para que cada noche se haga un rociado de agua de 5 minutos. Los primeros días, Luis no se fiaba y ha ido a verificar que las rutinas funcionaban bien, pero a partir del tercer día, ha visto que ya no necesitaba ir durante la semana, teniendo así más tiempo para realizar otras labores en casa. Además, para evitar perder la configuración, se ha hecho una copia de seguridad de todos los parámetros, evitando así tener que volver a reconfigurarlo todo si hay alguna incidencia.

Escenario 2

Carlos y el informático que tiene contratado, se han reunido hoy con todos los agricultores de la cooperativa para informarles que van a ofrecer un nuevo servicio de riego automático para todos ellos de manera gratuita.

El informático, ya ha instalado el servidor central en un ordenador que tienen en la oficina y explica a todos los agricultores los pasos a seguir para hacer uso de este. Además, ofrecerá soporte técnico a todos los que lo necesiten.

Primero, les hace instalar a todos la aplicación, les pasa los parámetros que deben poner y les hace registrarse. Una vez registrados, Carlos les reparte un pack de bienvenida que contiene un par de Raspberry Pi y unos cuantos sensores de temperatura, humedad y electroválvulas.

Los agricultores, cuando llegan a sus instalaciones montan el hardware que han recibido siguiendo un sencillo tutorial de conexión y empiezan a usar la aplicación.

Uno de ellos añade las dos placas que ha recibido y registra todos los sensores. A continuación, empieza a definir un seguido de rutinas, en función de la temperatura y la humedad. Además, con la aplicación recibe notificaciones cada vez que se lanza la ejecución de un programa. De esta manera, puede dedicar el tiempo a cultivar otras cosas mientras parte de su conreo se riega automáticamente.

Carlos, como coordinador, pide a todos los compañeros que realicen una copia de las configuraciones y se ofrece a guardarlas él, teniendo así, todos los datos por si algún día sucede algún imprevisto.

Escenario 3
Teresa tiene libre el fin de semana y decide replantar todas las flores que se le han secado. Ha visto que hay un sistema gratuito que le permite configurarse un sistema de riego automático, para no olvidarse nunca más de regar sus flores y ha decidido comprar por internet dos Raspberry Pi y un pack de sensores. Como el jardín no es muy grande, ha comprado una sola electroválvula. Durante la tarde, se lee el manual y consigue arrancar el servidor en una de las Raspberry Pi que ha comprado. Una vez ha terminado, se instala la aplicación, se registra en el sistema y configura la otra placa. En ella, conecta los sensores y actuadores donde el manual indica. Seguidamente se autentica, registra la placa, añade los dispositivos y empieza a probar rutinas de riego. Cuando verifica que todo funciona correctamente, se configura una rutina de riego de un par de veces por semana. El lunes cuando va a trabajar, recibe una notificación cuando el sistema empieza a regar, pero como esta reunida, no lo ve hasta la hora de comer. Mientras vuelve a casa con autobús, aprovecha para analizar la temperatura de la tierra y empieza a pensar en posibles bulbos que puede plantar en un futuro.

3.2 Flujos de interacción

Una vez tenemos identificados los perfiles de usuario y los escenarios de uso, se ha creado un diagrama de flujo para comprender como se realizará la interacción con la aplicación.

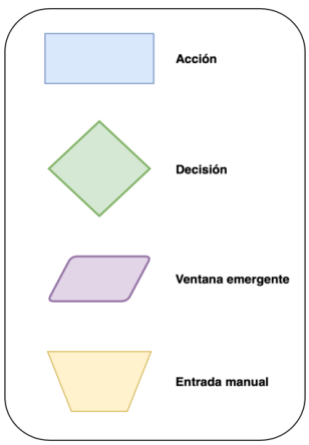


Figura 5: Leyenda del diagrama de flujo

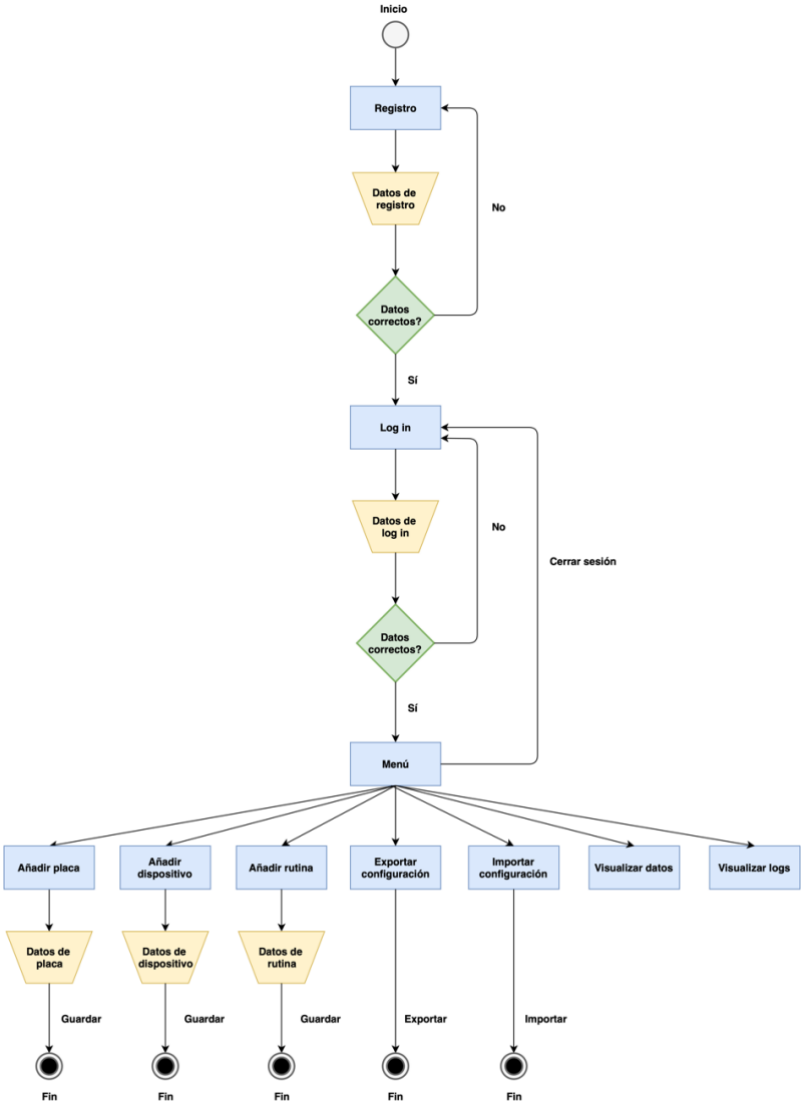


Figura 6: Flujo de interacción de la aplicación

Primeramente, se debe registrar el usuario en el sistema e iniciar sesión en él. A continuación, con la sesión activa en el menú principal de la aplicación, se podrán realizar diferentes acciones.

El orden recomendado de interacción es el siguiente:

1) Primero se añade una placa al sistema.

2) Seguidamente se añade un dispositivo a una placa existente.

3) A continuación, si existe un actuador, se configura una rutina sobre este.

4) Si se desea, se puede exportar la configuración realizada o importar una existente.

5) Por último, con todo el sistema configurado, se pueden visualizar los datos del sistema y ver los logs que se generen.

3.3 Diagramas de navegación

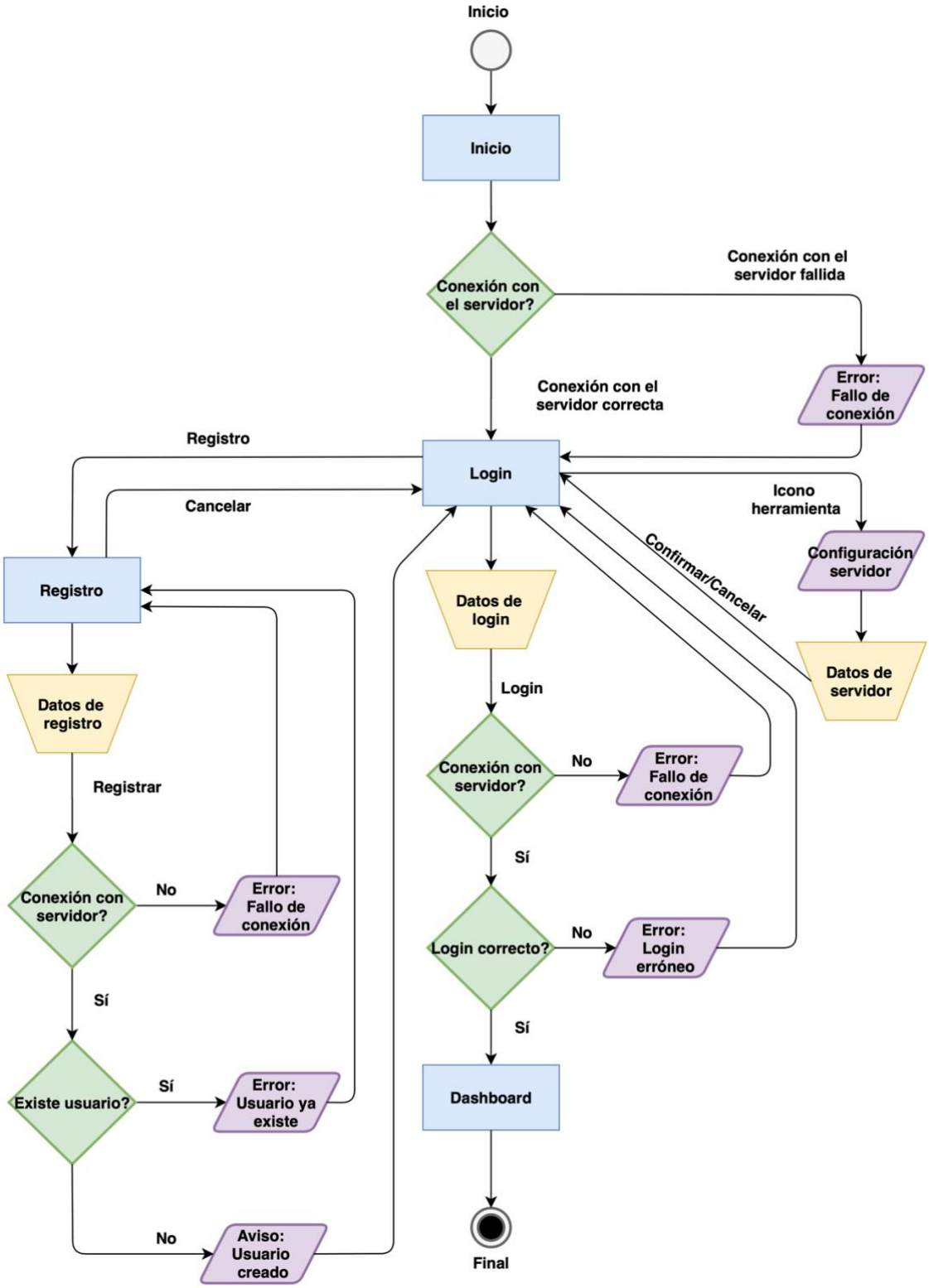


Figura 7: Diagrama del proceso de inicial de la aplicación

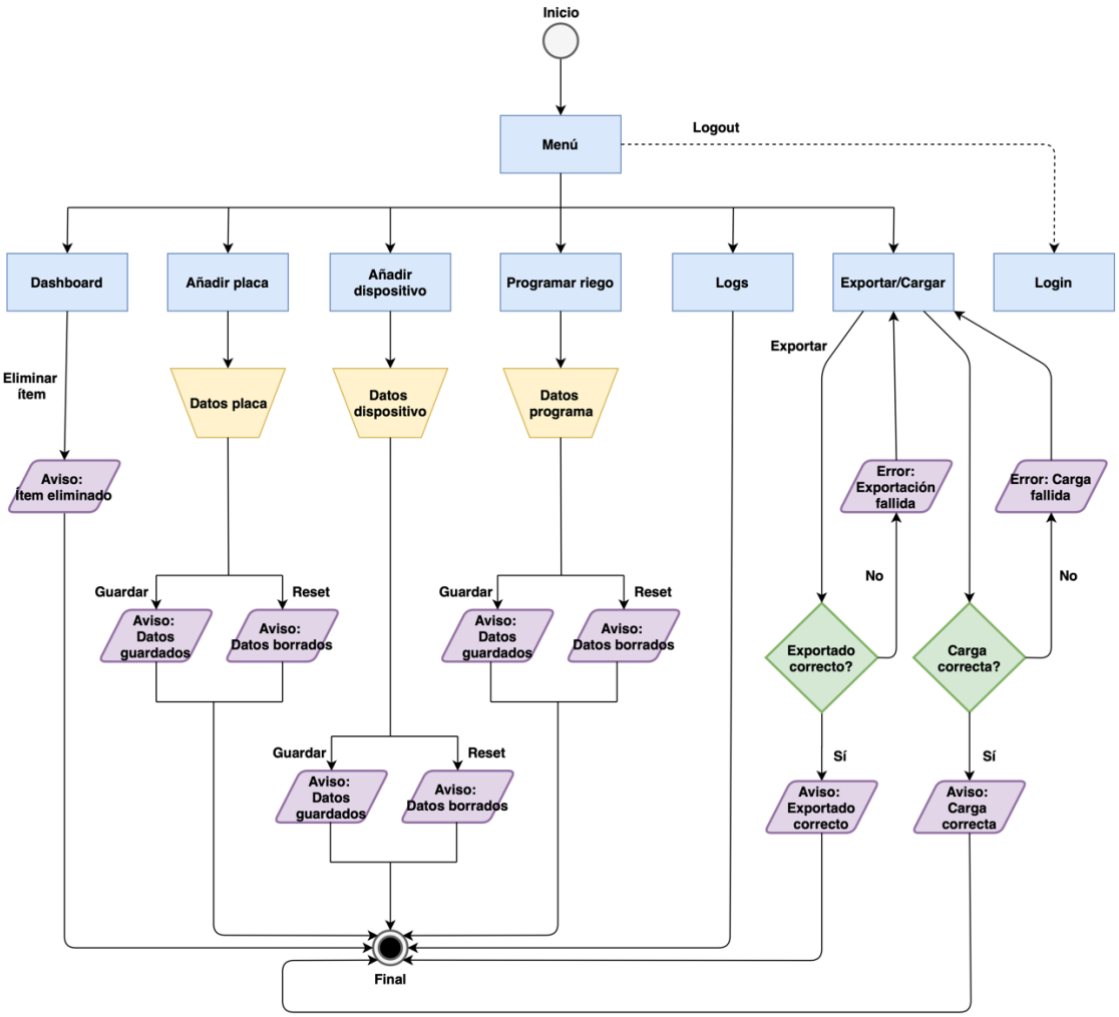


Figura 8: Diagrama de la navegación principal de la aplicación

Como podemos observar, en la Figura 7 se muestra como se realiza todo el proceso de configuración inicial y de autenticación. En la Figura 8, se muestra todo el flujo de navegación por el menú y por las diferentes pantallas de la aplicación.

4. Prototipado

En esta fase de prototipado se tiene como objetivo, realizar una aproximación al diseño de las interfaces que tendrá el sistema.

Partiendo de los flujos y diagramas de interacciones planteados anteriormente, se ha procedido a diseñar la aplicación de la siguiente manera:

- Se realizará un diseño siguiendo el patrón master-detail, es decir, una vez autenticados partiremos de un menú que en dispositivos móviles nos conducirá a la pantalla correspondiente y en tabletas, nos permitirá visualizar directamente el contenido de cada ítem.

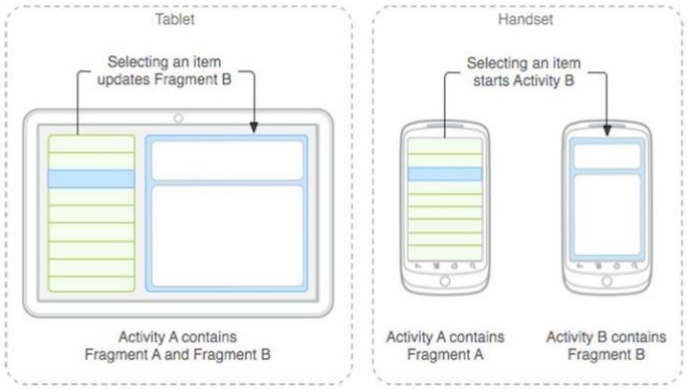


Figura 9: Esquema del patrón master-detail

  En una primera fase de diseño, se planteó usar una hamburguesa lateral con las diferentes opciones, pero al tener una navegación de estilo árbol, donde cada acción tiene un condicionante en la anterior, se ha visto que este diseño es mas fluido para el usuario ya que, en un menú lateral, las acciones suelen ser independientes.

- El menú principal incluirá las pantallas principales, que formarán la aplicación. Cada pantalla permitirá volver al menú principal con una opción de atrás, situada en la barra de navegación superior.

- Se ha decidido que las pantallas, serán independientes por cada acción que se pueda realizar sobre el sistema, y permitiendo que se tenga un espacio para visualizar los datos recogidos por este.

  Las interfaces diseñadas para cumplir con la función de la aplicación serán las siguientes:

  - Pantalla para añadir placa
  - Pantalla para añadir dispositivo
  - Pantalla para añadir rutina
  - Pantalla para importar o exportar configuración
  - Pantalla para visualizar datos
  - Pantalla para visualizar logs

- Las pantallas de registro y login, incorporaran una barra de navegación superior con una opción para configurar el servidor central y otra opción para visualizar un pequeño tutorial del funcionamiento y configuración del sistema.

- Cada pantalla incorporará una opción superior de información, que permitirá conocer para que sirve esa pantalla y como se debe utilizar correctamente.

- En todos los campos donde las opciones visibles estén condicionadas a una configuración anterior, existirá una opción de información, que permitirá al usuario entender porque no aparece el ítem esperado en el desplegable seleccionado.

4.1 Sketches

En la Figura 10 se muestran las diferentes pantallas de la aplicación, realizadas a mano alzada:

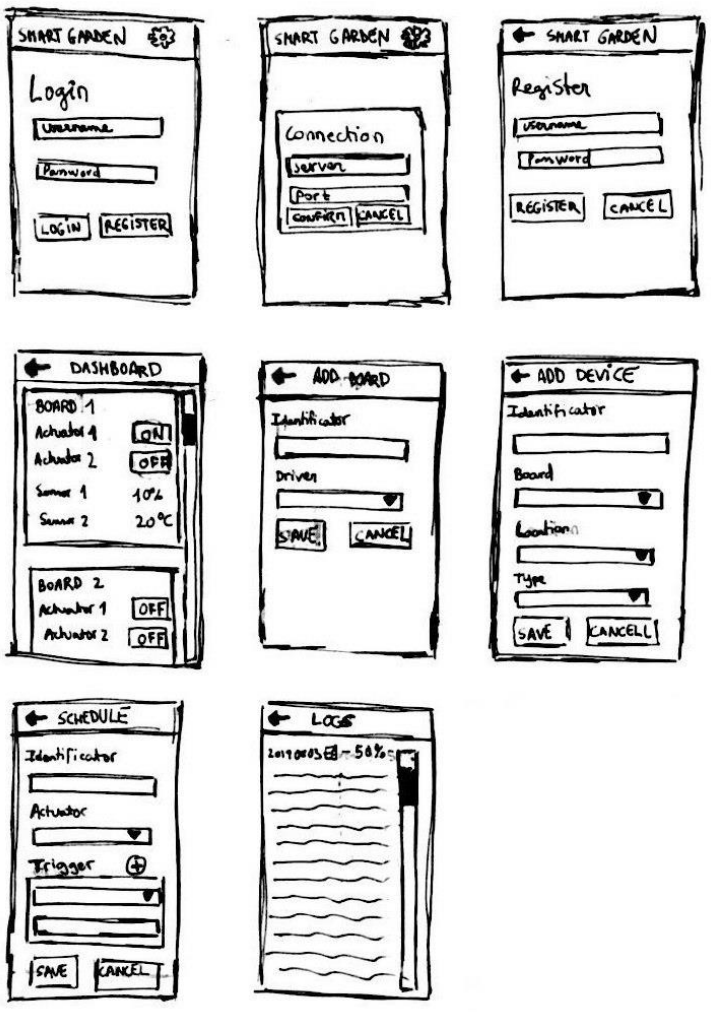


Figura 10: Sketches de la aplicación

4.2 Prototipo

Este apartado, se dividirá en cada una de las pantallas que formarán parte de la aplicación, junto con una descripción y una explicación de las diferentes funcionalidades de cada una.

4.2.1 Pantalla de inicio de sesión

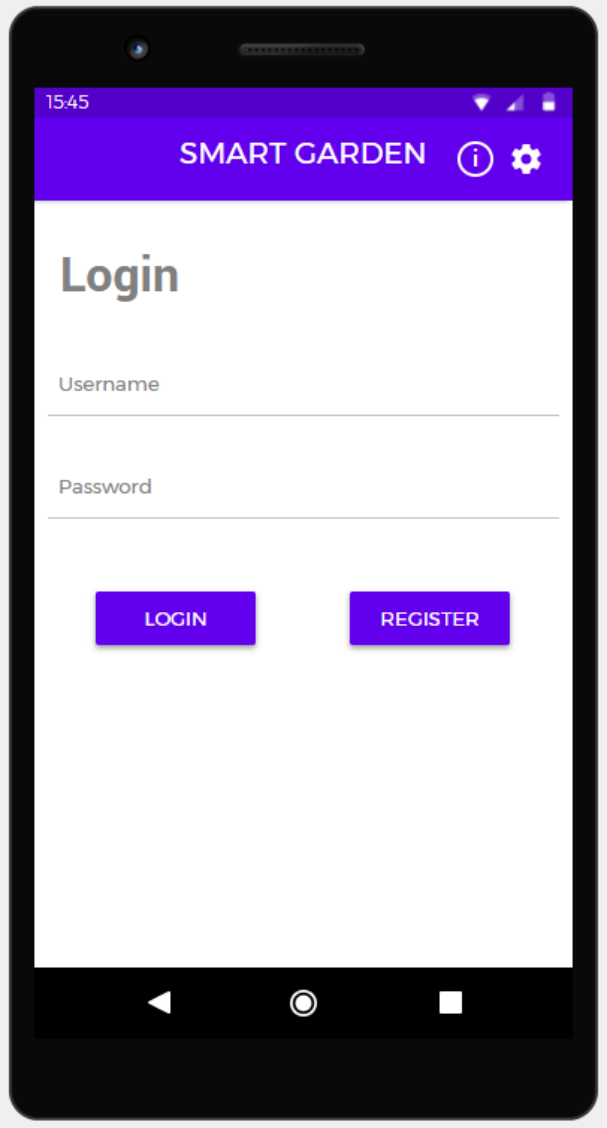


Figura 11: Prototipo de la pantalla de inicio de sesión

En la pantalla de inicio de sesión, es posible autenticarse con el nombre de usuario y contraseña utilizada para el registro. Una vez introducidos los datos, pulsando el botón Login, se procederá a iniciar la sesión. Si se pulsa el botón de configuración superior, el usuario podrá entrar en el dialogo de configuración del servidor. Pulsando el botón Registrar, el usuario accederá a la pantalla de registro.

4.2.2 Pantalla de registro

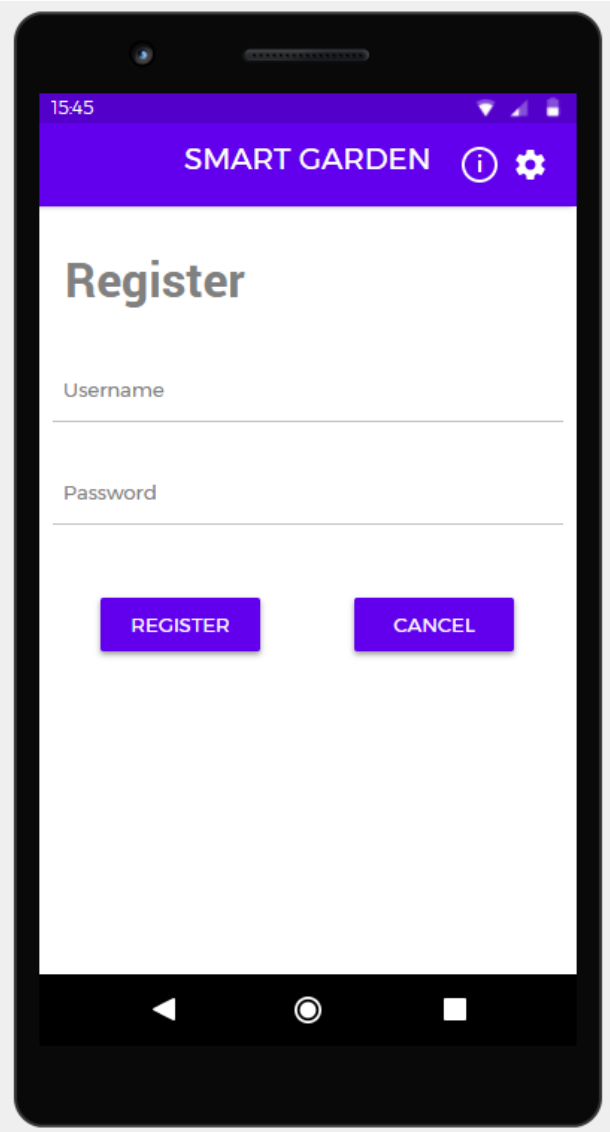


Figura 12: Prototipo de la pantalla de registro

En la pantalla de registro, es posible registrar un usuario en el sistema introduciendo un nombre de usuario, una contraseña y pulsando el botón Registrar. Pulsando el botón Cancelar, se vuelve a la pantalla de inicio de sesión. Pulsando el botón herramienta superior, el usuario podrá entrar en el dialogo de configuración del servidor.

4.2.3 Dialogo de configuración del servidor

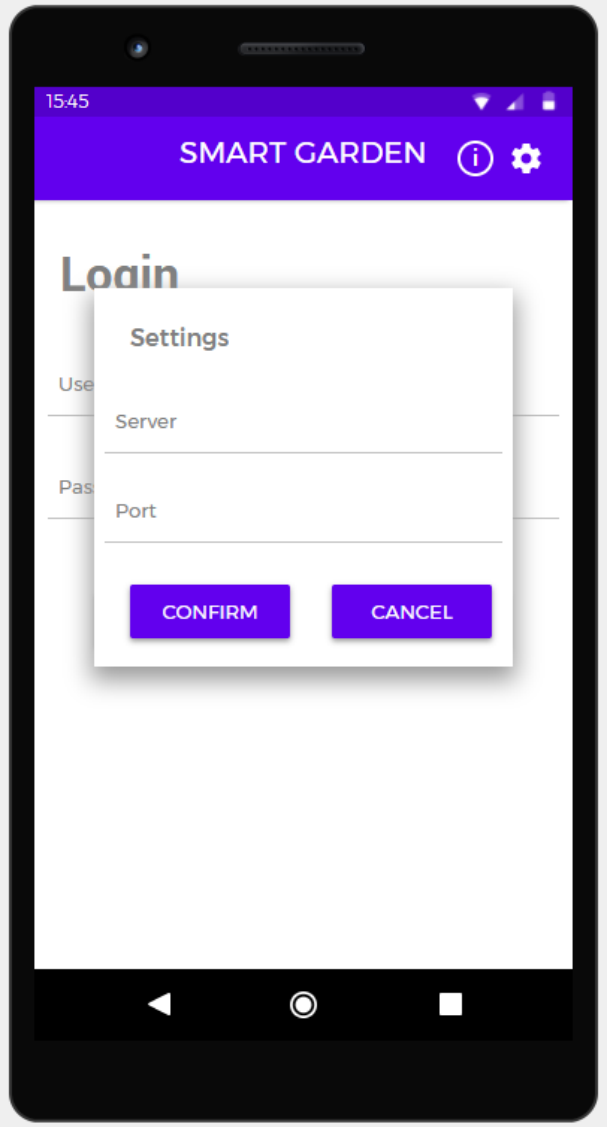


Figura 13: Prototipo del dialogo de configuración del servidor

En el dialogo de configuración del servidor, es posible introducir la dirección IP y el puerto donde se encuentra el servidor que apuntará la aplicación. Una vez introducidos los datos y pulsado el botón Confirmar, los datos quedan almacenados. El botón cancelar, permite volver a la pantalla desde donde hemos llamado este dialogo.

4.2.4 Menú principal

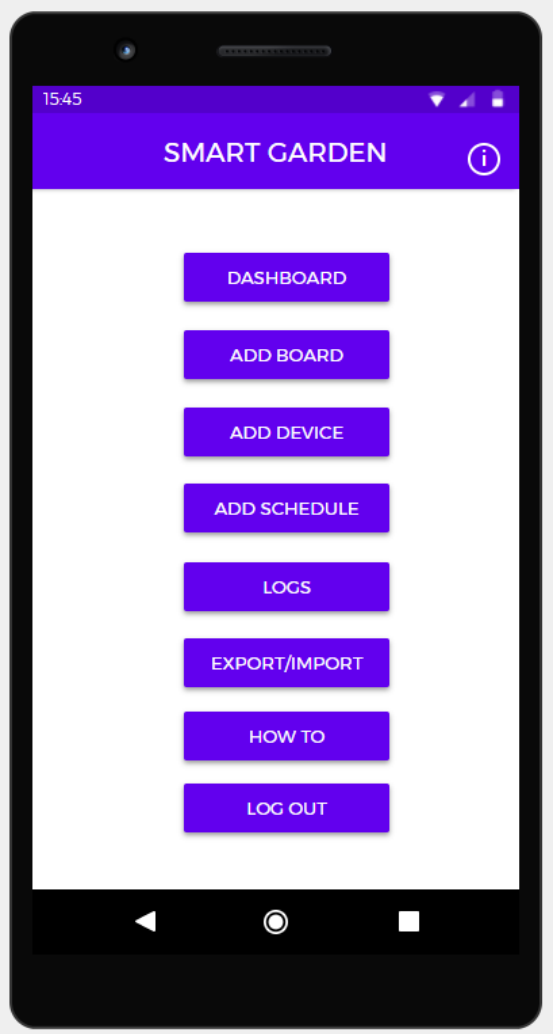


Figura 14: Prototipo del menú principal

El menú lateral permite navegar por las diversas pantallas que forman la aplicación:

- **Dashboard:** permite visualizar la pantalla principal de la aplicación.
- **Añadir placa:** permite visualizar la pantalla para añadir placas al sistema.
- **Añadir dispositivo:** permite visualizar la pantalla para añadir dispositivos al sistema.
- **Añadir planificación:** permite visualizar la pantalla para añadir rutinas al sistema.
- **Logs:** permite visualizar el registro de eventos del sistema.
- **Exportar/Importar:** permite exportar o importar un fichero de configuración.
- **How to:** permite visualizar un tutorial de como utilizar el sistema.
- **Log out:** permite cerrar la sesión en curso.

4.2.5 Pantalla principal (Dashboard)

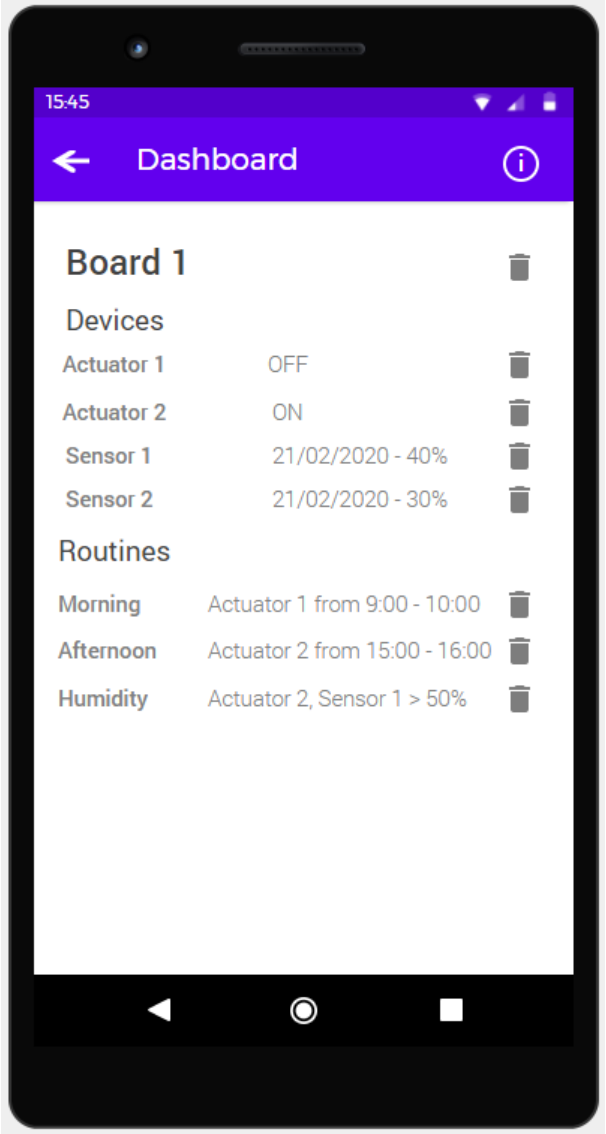


Figura 15: Prototipo de la pantalla dashboard

La pantalla principal permite ver las placas instaladas por el usuario en curso, así como sus dispositivos asociados y las rutinas instaladas. Presionando sobre el botón de eliminar de cada elemento, podemos eliminar del sistema ese ítem. Al lado de cada sensor podemos visualizar el último dato adquirido y la fecha de obtención. En el caso de los actuadores podemos visualizar su estado (encendido o apagado). En las rutinas podemos visualizar la configuración instalada en el equipo.

4.2.6 Pantalla para añadir placa

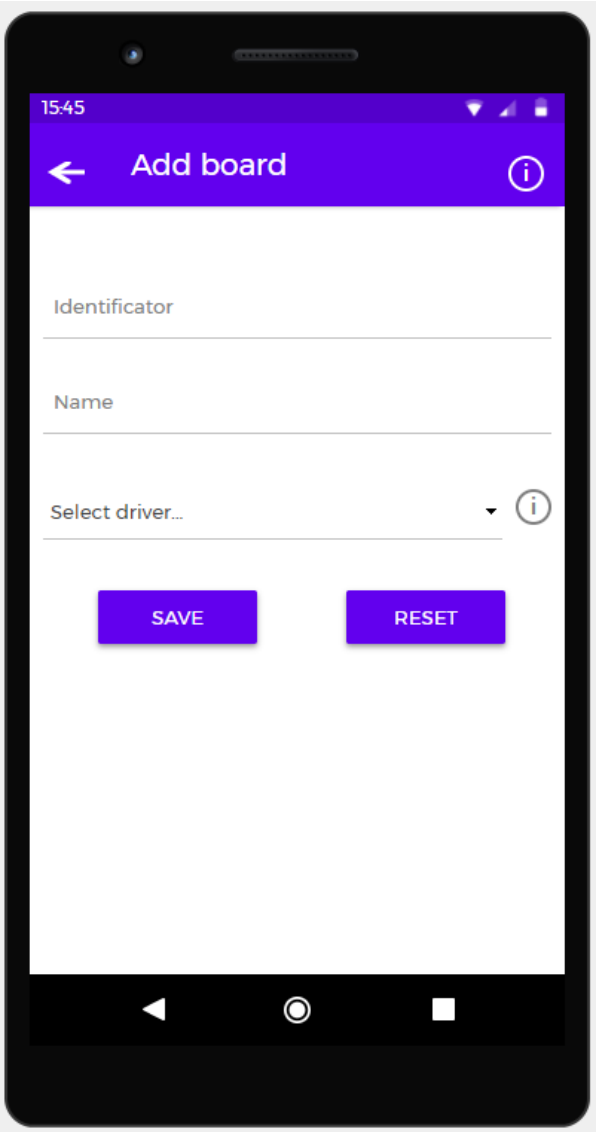


Figura 16: Prototipo de la pantalla para añadir placa

La pantalla de añadir placa, permite introducir un nombre descriptivo para la placa y seleccionar el controlador especifico al hardware que queremos instalar. También, permite visualizar el identificador interno que se le asignara a la nueva placa.

Presionando el botón Guardar, se guardarán los cambios. Presionando Reset, se generará un nuevo identificador y se borrarán los datos introducidos por el usuario.

4.2.7 Pantalla para añadir dispositivos

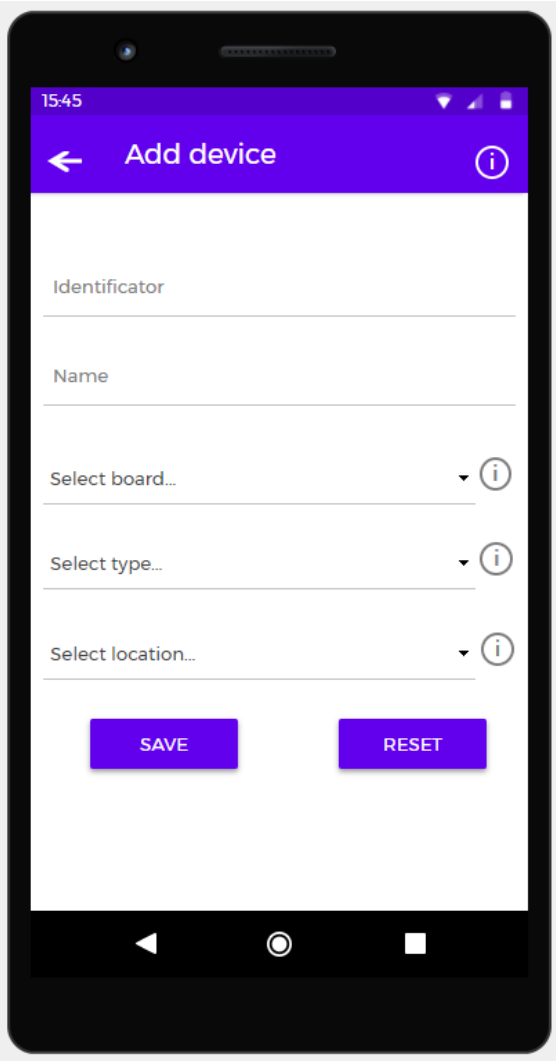


Figura 17: Prototipo de la pantalla para añadir dispositivos

La pantalla de añadir dispositivo permite introducir un nombre descriptivo para el dispositivo, la placa sobre la cual queremos instalar el dispositivo, el tipo de dispositivo a añadir (sensor o actuador) y la localización física de este dispositivo sobre la placa. También, permite visualizar el identificador interno que se le asignara al nuevo dispositivo.

Para seleccionar la placa, tenemos un desplegable donde se podrán visualizar todas las placas instaladas. En cuanto al desplegable de localización, veremos los pines disponibles en función del controlador seleccionado para cada placa.

Presionando Guardar, se salvarán todos los cambios. Presionando Reset, se borrarán todos los cambios introducidos y se generara un nuevo identificador.

4.2.8 Pantalla para programar rutinas

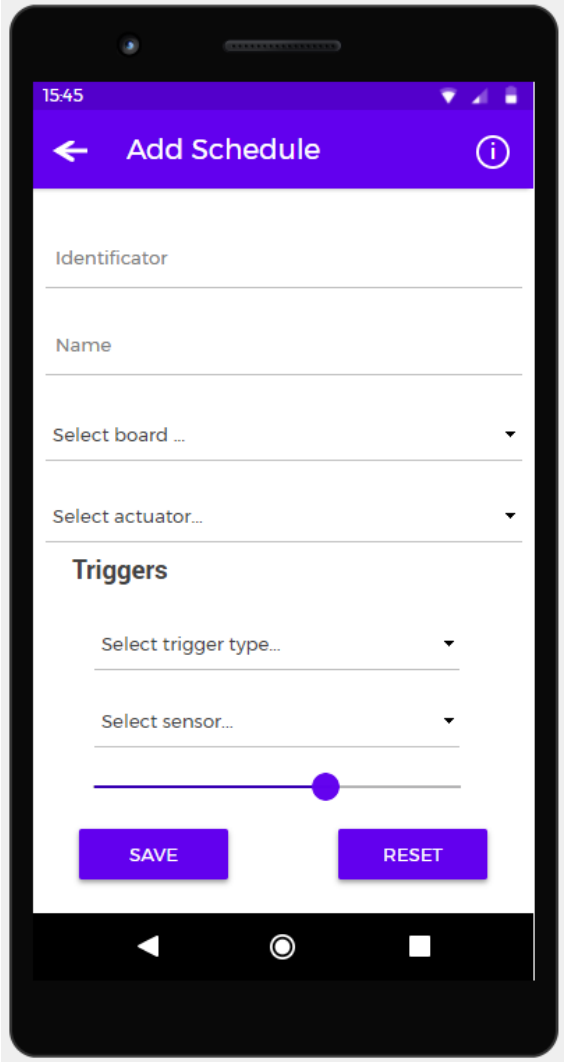


Figura 18: Prototipo de la pantalla para programar rutinas

La pantalla de programación de rutinas permite introducir un nombre descriptivo para la rutina, seleccionar que actuador va a ejecutarla y añadir diferentes disparadores para esta. También, permite visualizar el identificador interno que se le asignara a la nueva rutina.

Para seleccionar el actuador, tenemos un desplegable donde se podrán visualizar todos los actuadores instalados. Cuando se inserte un disparador, podremos seleccionar el tipo (hora de inicio - hora final o evento). En función del tipo de disparador, tendremos la opción de seleccionar el valor o un seleccionador de fecha. También tendremos un desplegable que nos permitirá seleccionar el sensor que queremos, en caso de que el disparador sea por evento.

Presionando Guardar, se salvarán todos los cambios. Presionando Reset, se borrarán todos los cambios introducidos y se generara un nuevo identificador.

4.2.9 Pantalla de logs

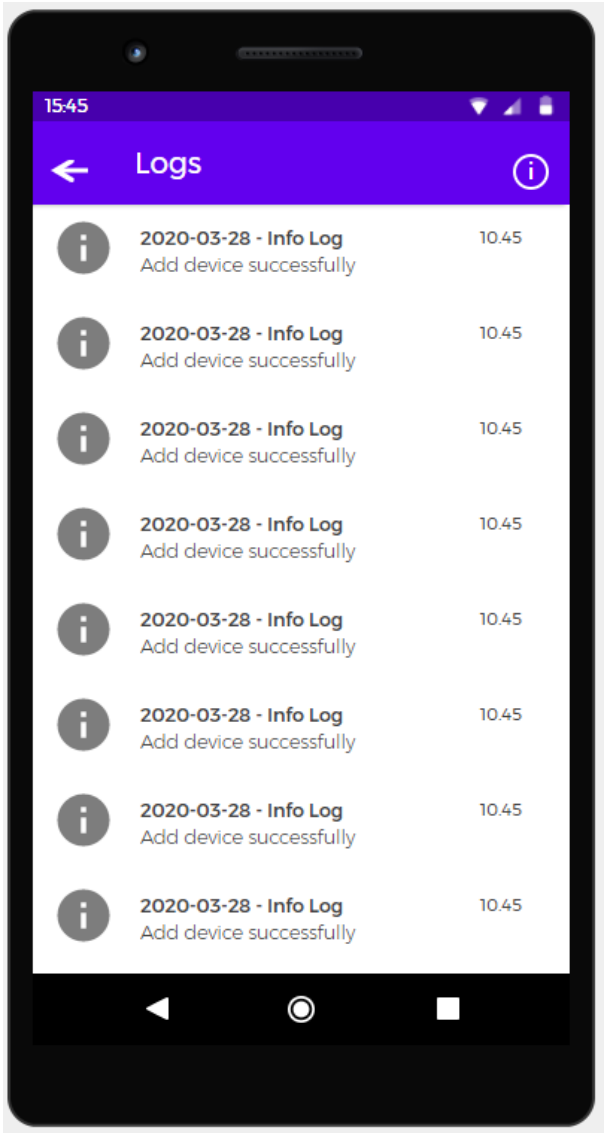


Figura 19: Prototipo de la pantalla de logs

La pantalla de logs muestra una lista de entradas ordenadas por fecha. Cada entrada contiene la fecha y el nivel de log al que pertenecen. También, incluyen una descripción del evento y del motivo por el cual ha sucedido.

5.2.10 Pantalla de exportación y carga

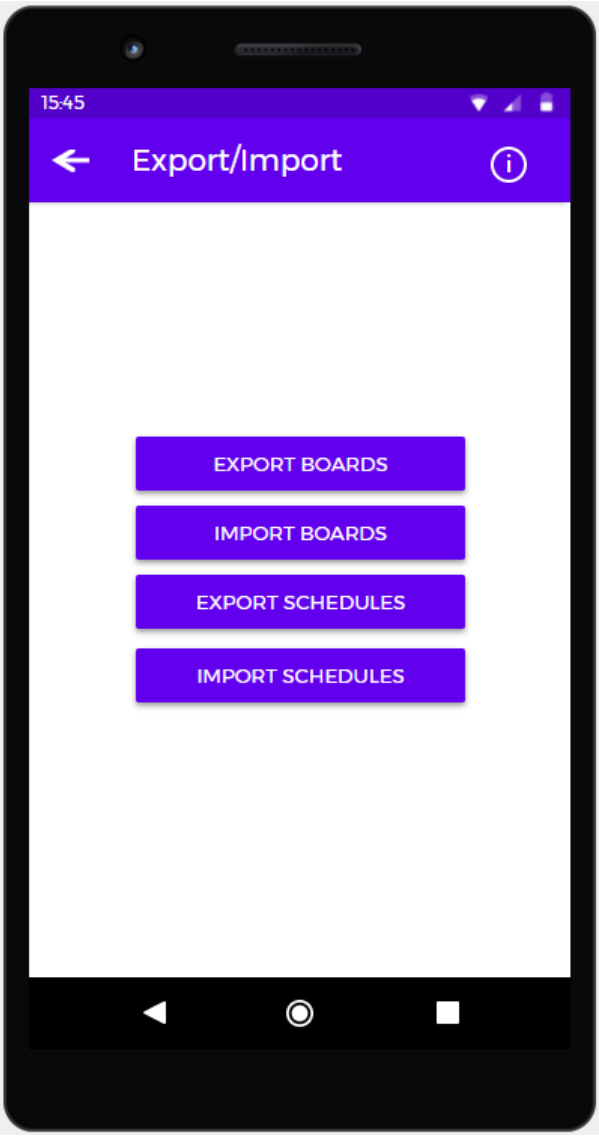


Figura 20: Prototipo de la pantalla de exportación y carga

La pantalla de exportación y importación permite seleccionar el directorio donde queremos exportar el fichero de placas y el fichero de rutinas.

Al seleccionar uno de los botones de exportación, aparecerá un navegador que nos permitirá fijar el nombre y el destino.

Al seleccionar uno de los botones de importación, aparecerá un navegador que nos permitirá seleccionar el fichero que queremos cargar.

5. Casos de uso

5.1 Registrar usuario

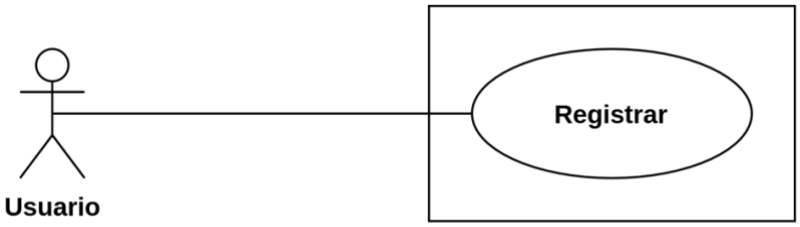


Figura 21: Casos de uso – Registrar

Actores
Cualquier usuario que utilice la aplicación.
Precondiciones
Tener instalado y arrancado el servidor con la base de datos.
Postcondiciones
Precondición 1: Se registra el usuario en el sistema.
Flujo principal
Paso 1: Abrir la aplicación. **Paso 2:** En la pantalla de inicio de sesión, pulsar el botón “Registrar”. **Paso 3:** Rellenar los campos nombre de usuario y contraseña. **Paso 4:** Pulsar el botón “Registrar”. **Paso 5:** Aparece un mensaje indicando que se ha registrado el usuario.
Flujo alternativo
Escenario alternativo 1: **Paso 4:** Pulsar el botón “Registrar”. **Paso 5:** Aparece un mensaje indicando que ya existe el usuario. **Paso 6:** Cambiar el nombre de usuario y repetir la operación. **Escenario alternativo 2:** **Paso 4:** Pulsar el botón “Registrar”. **Paso 5:** Aparece un mensaje indicando que la conexión con el servidor ha fallado. **Paso 6:** Cambiar configuración del servidor y repetir la operación.

Escenario alternativo 3: **Paso 4:** Pulsar el botón “Cancelar”. **Paso 5:** Se vuelve a la pantalla anterior.

5.2 Configurar el servidor

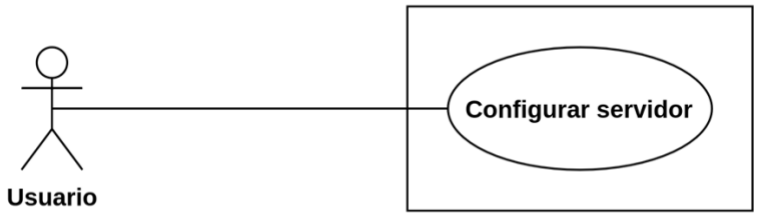


Figura 22: Casos de uso - Configurar servidor

Actores
Cualquier usuario que utilice la aplicación.
Precondiciones
Precondición 1: Tener instalado y arrancado el servidor con la base de datos.
Postcondiciones
Se configura la IP y el puerto del servidor.
Flujo principal
Paso 1: Abrir la aplicación. **Paso 2:** En la pantalla de inicio de sesión, pulsar botón “Herramienta”. **Paso 3:** Rellenar los campos IP y puerto. **Paso 4:** Pulsar el botón “Confirmar”. **Paso 5:** Se vuelve a la pantalla anterior.
Flujo alternativo
Escenario alternativo 1: **Paso 4:** Pulsar el botón “Cancelar”. **Paso 5:** Se vuelve a la pantalla anterior.

5.3 Autenticar usuario

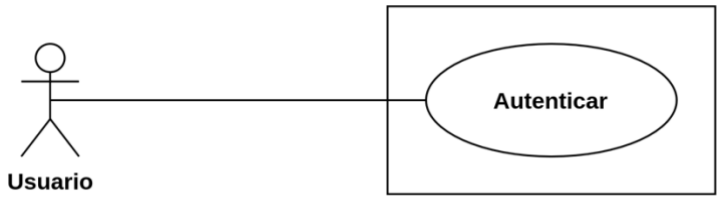


Figura 23: Casos de uso – Autenticar

Actores
Cualquier usuario que utilice la aplicación.
Precondiciones
Precondición 1: Tener instalado y arrancado el servidor con la base de datos **Precondición 2:** Estar registrado en el servidor.
Postcondiciones
Proporcionar unas credenciales válidas al sistema.
Flujo principal
Paso 1: Abrir la aplicación. **Paso 2:** En la pantalla de inicio de sesión, rellenar las credenciales. **Paso 3:** Pulsar el botón “Log in” **Paso 4:** Aparece la pantalla siguiente.
Flujo alternativo
Escenario alternativo 1: **Paso 3:** Pulsar el botón “Log in”. **Paso 4:** Aparece un mensaje indicando que las credenciales son erróneas. **Paso 5:** Cambiar las credenciales y repetir la operación. **Escenario alternativo 2:** **Paso 3:** Pulsar el botón “Log in”. **Paso 4:** Aparece un mensaje indicando que la conexión con el servidor ha fallado. **Paso 5:** Cambiar configuración del servidor y repetir la operación.

5.4 Configurar el sistema

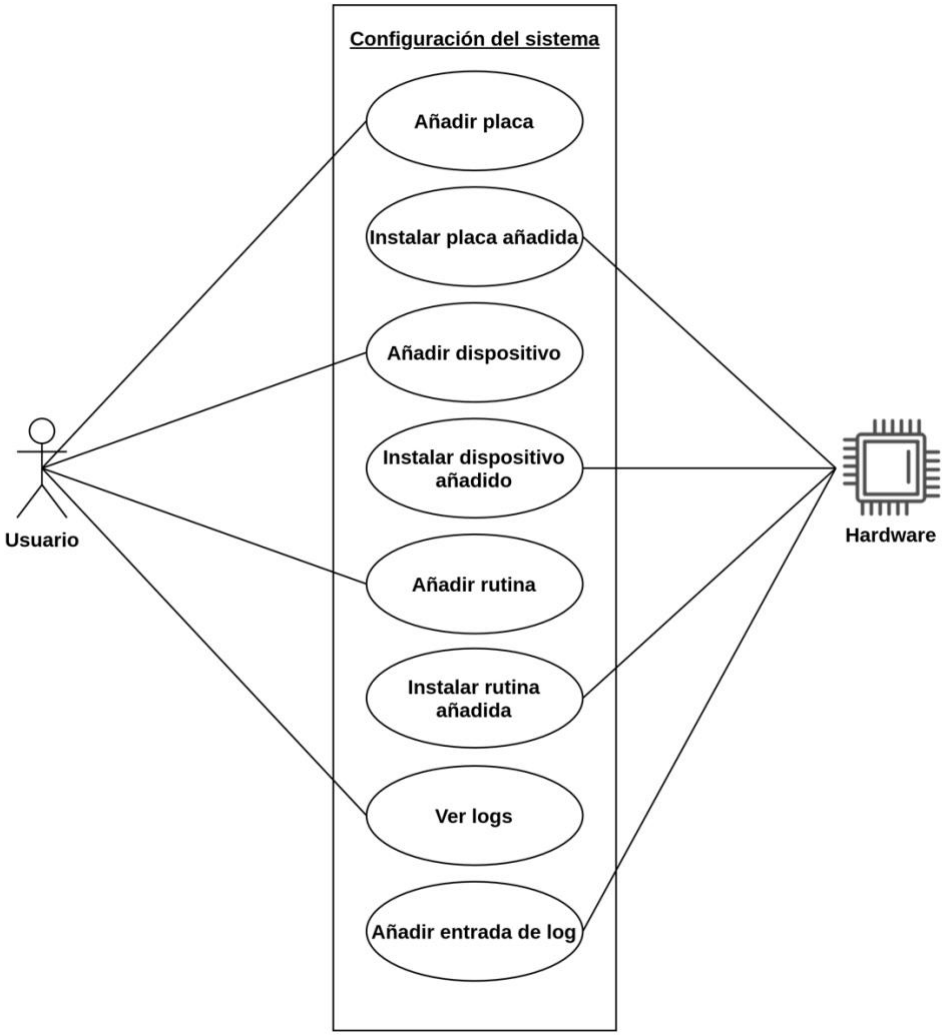


Figura 24: Casos de uso - Configurar el sistema

Actores
Actor 1: Cualquier usuario que utilice la aplicación. **Actor 2:** Cualquier equipo Raspberry Pi configurado en el sistema.
Precondiciones
Precondición 1: Tener instalado y arrancado el servidor con la base de datos **Precondición 2:** Tener un equipo Raspberry Pi configurado en el sistema. **Precondición 3:** Estar autenticado en la aplicación.
Postcondiciones
Instalar placas, dispositivos, rutinas y visualizar logs del sistema.

Flujo principal “Añadir placa”
Paso 1: En el menú lateral seleccionar “Añadir placa”. **Paso 2:** En la pantalla de añadir placa rellenar los campos necesarios. **Paso 3:** Pulsar el botón “Guardar” **Paso 4:** Aparece un mensaje indicando que se ha añadido la placa.
Flujo alternativo “Añadir placa”
Escenario alternativo 1: **Paso 3:** Pulsar el botón “Reset” **Paso 4:** Aparece un mensaje indicando que los datos han sido eliminados.
Flujo principal “Instalar placa añadida”
El dispositivo Raspberry Pi ve los datos de la placa añadida por el usuario.
Flujo principal “Añadir dispositivo”
Paso 1: En el menú lateral seleccionar “Añadir dispositivo”. **Paso 2:** En la pantalla de añadir dispositivo rellenar los campos necesarios. **Paso 3:** Pulsar el botón “Guardar”. **Paso 4:** Aparece un mensaje indicando que se ha añadido el dispositivo.
Flujo alternativo “Añadir dispositivo”
Escenario alternativo 1: **Paso 3:** Pulsar el botón “Reset” **Paso 4:** Aparece un mensaje indicando que los datos han sido eliminados.
Flujo principal “Instalar dispositivo añadido”
El dispositivo Raspberry Pi ve los datos del dispositivo añadido por el usuario.
Flujo principal “Añadir rutina”
Paso 1: En el menú lateral seleccionar “Programar riego”. **Paso 2:** En la pantalla de programar riego rellenar los campos necesarios. **Paso 3:** Pulsar el botón “Guardar”. **Paso 4:** Aparece un mensaje indicando que se ha añadido la rutina.
Flujo alternativo “Añadir rutina”
Escenario alternativo 1: **Paso 3:** Pulsar el botón “Reset” **Paso 4:** Aparece un mensaje indicando que los datos han sido eliminados.

Flujo principal “Instalar rutina añadida”
El dispositivo Raspberry Pi ve los datos de la rutina añadida por el usuario.
Flujo principal “Ver logs”
Paso 1: En el menú lateral seleccionar “Logs”. **Paso 2:** En la pantalla de logs se visualizan los datos de información almacenados en el sistema.
Flujo principal “Añadir entrada de log”
El dispositivo Raspberry Pi inserta datos de log para ser visualizados por el usuario.

5.5 Manipulación de datos

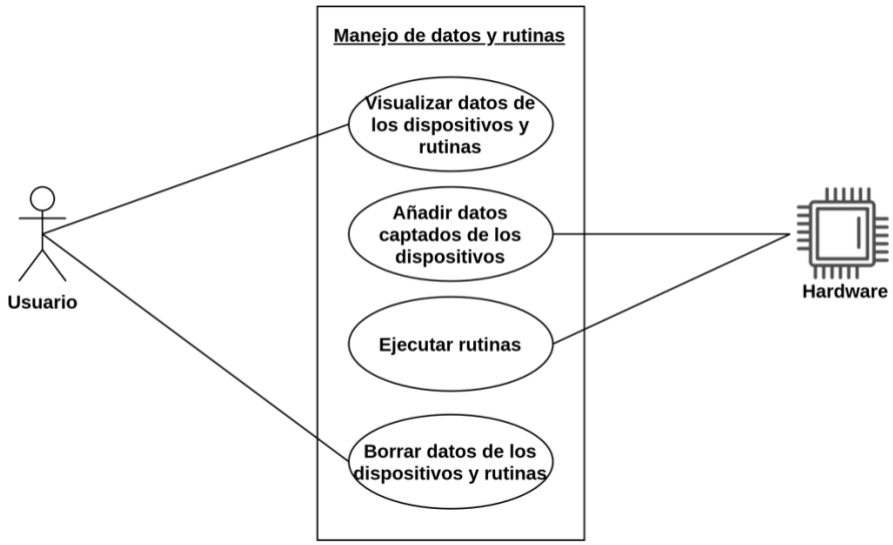


Figura 25: Casos de uso - Manipulación de datos

Actores
Actor 1: Cualquier usuario que utilice la aplicación. **Actor 2:** Cualquier equipo Raspberry Pi configurado en el sistema.
Precondiciones
Precondición 1: Tener instalado y arrancado el servidor con la base de datos **Precondición 2:** Tener un equipo Raspberry Pi configurado en el sistema. **Precondición 3:** Estar autenticado en la aplicación.

Postcondiciones
Visualizar datos y gestionar los dispositivos y rutinas.
Flujo principal “Visualizar datos de los dispositivos y rutinas”
Paso 1: En el menú lateral seleccionar “Dashboard”. **Paso 2:** En la pantalla de Dashboard visualizar los datos del sistema.
Flujo principal “Añadir datos captados de los dispositivos”
El dispositivo Raspberry Pi envía datos de los sensores y actuadores instalados.
Flujo principal “Ejecutar rutinas”
El dispositivo Raspbery Pi envía datos de las rutinas instaladas.
Flujo principal “Borrar datos de los dispositivos y rutinas”
Paso 1: En el menú lateral seleccionar “Dashboard”. **Paso 2:** En la pantalla de Dashboard, pulsar el botón eliminar de la placa, dispositivo o rutina a eliminar. **Paso 3:** Aparece un mensaje indicando que el ítem ha sido eliminado.

5.6 Exportación y carga de configuración

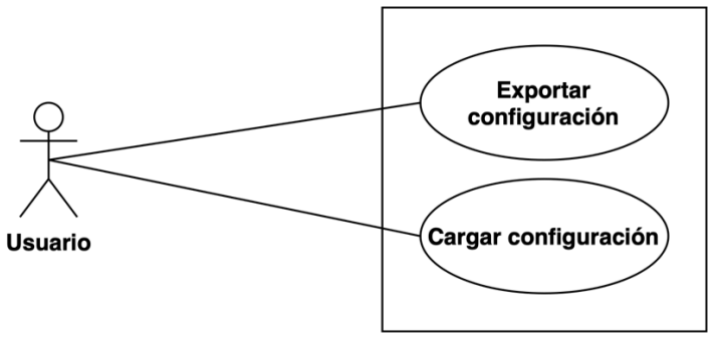


Figura 26: Casos de uso - Exportación y carga de configuración

Actores
Cualquier usuario que utilice la aplicación.
Precondiciones
Precondición 1: Tener instalado y arrancado el servidor con la base de datos **Precondición 2:** Estar registrado en el servidor. **Precondición 3:** Tener alguna configuración instalada.

Postcondiciones
Exportar y importar un fichero de configuración.
Flujo principal “Exportar configuración”
Paso 1: En el menú lateral seleccionar “Exportar/ Importar”. **Paso 2:** En la pantalla de “Exportar/ Importar” presionar el botón “Exportar”. **Paso 3:** Aparece un mensaje indicando que se ha exportado la configuración correctamente.
Flujo alternativo “Exportar configuración”
Escenario alternativo 1 **Paso 2:** En la pantalla de “Exportar/ Importar” presionar el botón “Exportar”. **Paso 3:** Aparece un mensaje indicando que la exportación ha fallado. **Paso 4:** Revisar configuración y permisos y repetir la operación.
Flujo principal “Importar configuración”
Paso 1: En el menú lateral seleccionar “Exportar/Importar”. **Paso 2:** En la pantalla de “Exportar/Importar” presionar el botón “Importar”. **Paso 3:** Seleccionar el fichero de configuración a importar. **Paso 4:** Aparece un mensaje indicando que se ha cargado la configuración correctamente.
Flujo alternativo “Importar configuración”
Escenario alternativo 1 **Paso 3:** Seleccionar el fichero de configuración a importar. **Paso 4:** Aparece un mensaje indicando que la carga ha fallado. **Paso 5:** Revisar el fichero de configuración y repetir la operación.

5.7 Uso global del sistema

Figura 27: Caso de uso - Sistema global

6. Arquitectura

6.1 Arquitectura del sistema

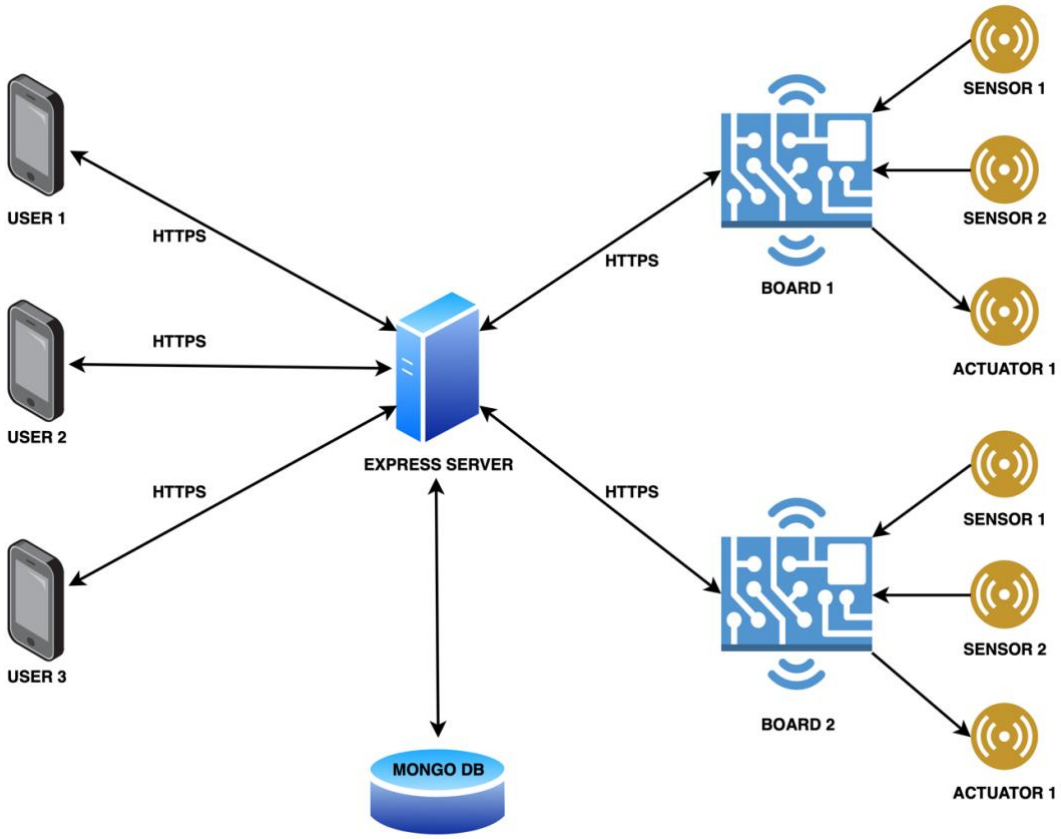


Figura 28: Diagrama general del sistema

El diagrama de la Figura 28 muestra los diferentes elementos de la arquitectura del sistema a alto nivel.

Podemos diferencias las siguientes partes:

- **Sensores y actuadores:** son los diferentes elementos hardware que se acoplan físicamente a las placas Raspberry Pi que tengamos instaladas. Se comunican con las placas mediante el protocolo 2-Wire sensor.

- **Placas de control:** son las diferentes placas Raspberry Pi que interactúan con los sensores y actuadores y a su vez, mediante el protocolo HTTPS, envían los datos de estado al servidor Express para su almacenaje.

  Para realizar todo este proceso, cuentan con un script de Python que el usuario debe poner en funcionamiento y que se encarga de aportar la inteligencia necesaria a las placas.

- **Servidor Express:** servidor que proporciona una API Rest para interactuar y formatear los datos de la base de datos MongoDB.

- **MongoDB:** base de datos no relacional que sirve de almacenamiento para todos los datos del sistema.

- **Usuarios:** clientes que utilizan la aplicación y se comunican mediante HTTPS con el servidor Express.

El diagrama de la Figura 29, muestra donde se ejecutará cada parte del sistema diseñado. Como podemos ver, el servidor Express y la base de datos MongoDB, se ejecutarán siempre en una única máquina, llamada servidor central. Por otro lado, los dispositivos móviles ejecutaran el cliente Android y las placas ejecutaran el cliente Python.

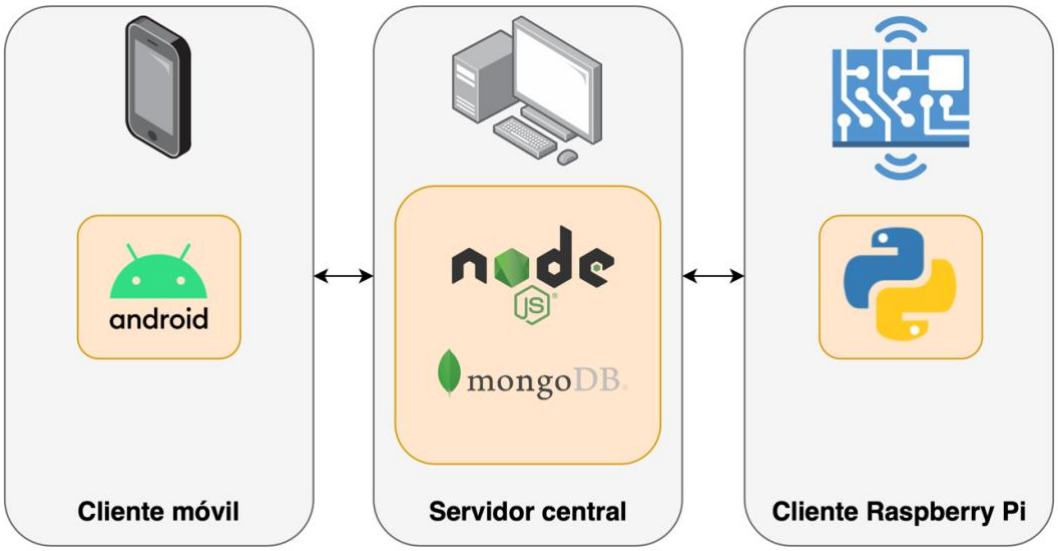


Figura 29: Organización del sistema

6.2. Arquitectura de la base de datos

La base de datos se ha estructurado siguiente el diagrama de clases de la Figura 30.

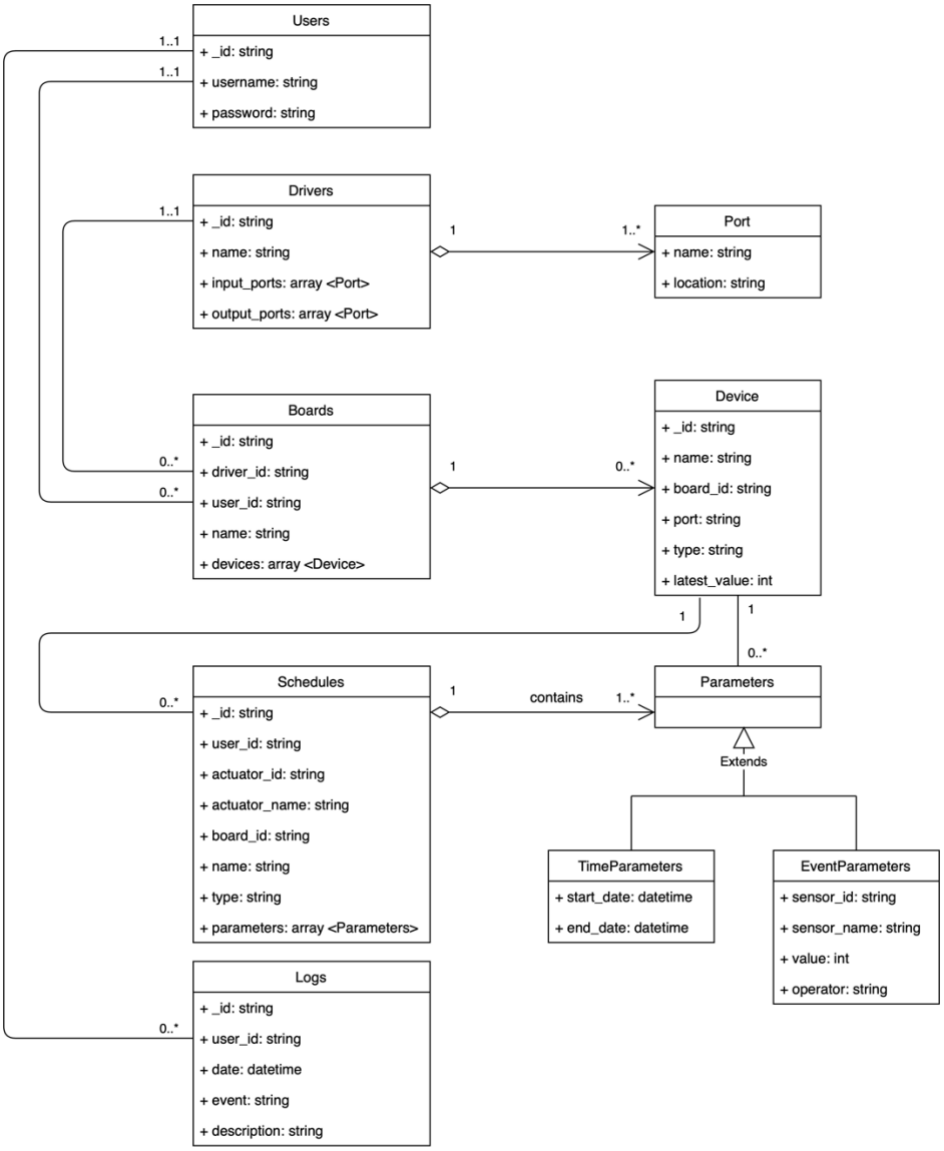


Figura 30: Arquitectura de la base de datos

Como trabajamos con una base de datos no relacional, tenemos las siguientes colecciones:

- **USERS:** se almacenan los datos relativos a los usuarios registrados.
- **DRIVERS:** se almacenan los datos de los controladores disponibles. Cada controlador tiene dos listas de elementos puerto.
- **BOARDS:** se almacenan los datos de cada placa y a que usuario pertenecen. Cada placa tiene una lista de dispositivos instalados.

- **SCHEDULES:** se almacenan las rutinas instaladas por cada actuador. Cada rutina tiene una lista de parámetros de configuración.

- **LOGS:** se almacenan las entradas de log de cada usuario.

6.3 Arquitectura de la API

Descripción	Tipo	URL
Devuelve si el usuario se ha identificado correctamente y el jwt token de sesión.	**POST**	**/login**
Devuelve si el usuario se ha registrado correctamente o si existe algún usuario con el mismo nombre.	**POST**	**/register**
Devuelve si el driver se ha añadido correctamente.	**POST**	**/addDriver**
Devuelve todos los drivers instalados en el sistema.	**GET**	**/drivers**
Devuelve el driver con identificador {id}.	**GET**	**/driver/{id}**
Devuelve si la placa se ha añadido correctamente.	**POST**	**/addBoard**
Devuelve si se ha podido eliminar correctamente la placa con identificador {id}.	**DELETE**	**/deleteBoard/{id}**
Devuelve todas las placas instaladas en el sistema para el usuario que hace la petición.	**GET**	**/boards**
Devuelve si el dispositivo se ha añadido correctamente.	**POST**	**/addDevice**
Devuelve si se ha podido eliminar correctamente el dispositivo con identificador {deviceId} de la placa {boardId}.	**DELETE**	**/deleteDevice/ {boardId}/{deviceId}**

Devuelve si se ha podido insertar el valor al dispositivo especificado en la petición.	**POST**	**/addDeviceValue**
Devuelve si la rutina se ha añadido correctamente.	**POST**	**/addSchedule**
Devuelve si se ha podido eliminar correctamente la rutina con identificador {id}.	**DELETE**	**/deleteSchedule/{id}**
Devuelve todas las rutinas instaladas en el sistema para el usuario que hace la petición.	**GET**	**/schedules**
Devuelve si el log se ha añadido correctamente.	**POST**	**/addLog**
Devuelve todos los logs existentes en el sistema para el usuario que hace la petición.	**GET**	**/logs**

Figura 31: Tabla de peticiones REST

Petición	Código	Descripción
/login	**200**	Inicio de sesión correcto.
/login	**401**	Credenciales invalidas.
/login	**500**	Error interno.
/register	**201**	Registro correcto.
/register	**409**	Usuario con el mismo nombre existente.
/register	**500**	Error interno.
/addDriver	**201**	Driver añadido correctamente.
/addDriver	**500**	Error interno.

/drivers	**200**	Lectura de drivers correcta.
/drivers	**401**	Datos de sesión inválidos.
/drivers	**500**	Error interno.
/driver/{id}	**200**	Lectura de driver correcta.
/driver/{id}	**401**	Datos de sesión inválidos.
/driver/{id}	**500**	Error interno.
/addBoard	**201**	Placa añadida correctamente.
/addBoard	**401**	Datos de sesión inválidos.
/addBoard	**409**	Id de placa duplicado.
/addBoard	**500**	Error interno.
/deleteBoard/{id}	**200**	Placa borrada correctamente
/deleteBoard/{id}	**401**	Datos de sesión inválidos.
/deleteBoard/{id}	**500**	Error interno.
/boards	**200**	Lectura de placas correcta.
/boards	**401**	Datos de sesión inválidos.
/boards	**500**	Error interno.
/addDevice	**201**	Dispositivo añadido correctamente.
/addDevice	**401**	Datos de sesión inválidos.
/addDevice	**409**	Id de dispositivo duplicado.

/addDevice	**500**	Error interno.
/deleteDevice/{id}	**200**	Dispositivo borrado correctamente.
/deleteDevice/{id}	**401**	Datos de sesión inválidos.
/deleteDevice/{id}	**500**	Error interno.
/addDeviceValue	**201**	Valor añadido correctamente.
/addDeviceValue	**401**	Datos de sesión inválidos.
/addDeviceValue	**500**	Error interno.
/addSchedule	**201**	Rutina añadida correctamente.
/addSchedule	**401**	Datos de sesión inválidos.
/addSchedule	**409**	Id de rutina duplicado.
/addSchedule	**500**	Error interno.
/deleteSchedule/{id}	**200**	Rutina borrada correctamente.
/deleteSchedule/{id}	**401**	Datos de sesión inválidos.
/deleteSchedule/{id}	**500**	Error interno.
/schedules	**200**	Lectura de rutinas correcta.
/schedules	**401**	Datos de sesión inválidos.
/schedules	**500**	Error interno.
/addLog	**201**	Log añadido correctamente.
/addLog	**401**	Datos de sesión inválidos.

/addLog	**500**	Error interno
/logs	**200**	Lectura de logs correcta.
/logs	**401**	Datos de sesión inválidos.
/logs	**500**	Error interno.

Figura 32: Tabla de códigos en las peticiones REST

6.4. Estructura del código

Para estructurar el código de la aplicación, se ha decidido dividir los paquetes en dos grupos, tal como vemos en la Figura 33 y en la Figura 34. De esta manera, se puede diferenciar el código referente a las comunicaciones y el código referente a las vistas de usuario.

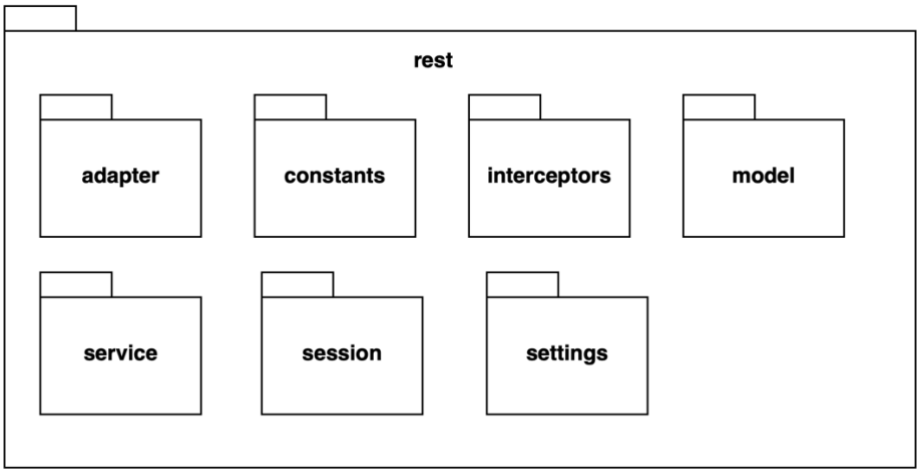


Figura 33: Paquete Rest

En el **paquete rest**, se han creado diferentes paquetes, en función de su finalidad.

- El **paquete adapter**, contiene los módulos necesarios para acceder a las peticiones que ofrece la API del servidor.
- El **paquete constants**, contiene las direcciones URL de la API y algunos valores útiles de retorno.
- El **paquete interceptors**, contiene los módulos para modificar las cabeceras y añadir el token JWT.

- El **paquete model**, contiene los diferentes objetos para serializar y des-serializar los datos JSON recibidos y enviados en las peticiones.

- El **paquete service**, contiene el servicio Rest para comunicarnos con el sevidor.

- El **paquete session**, contiene los módulos para almacenar en el dispositivo el id de usuario y el token de sesión.

- El **paquete settings**, contiene los módulos para almacenar en el dispositivo los parámetros de red configurados.

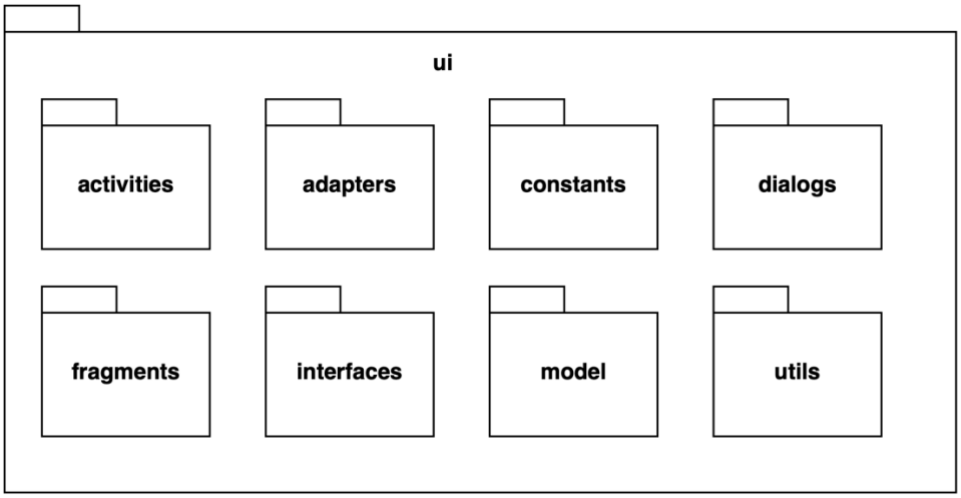


Figura 34: Paquete UI

En el **paquete ui**, se han creado diferentes paquetes, en función de su finalidad.

- El **paquete activities**, contiene las activities principales de la aplicación.

- El **paquete adapters**, contiene los diferentes adapters usados en distintos puntos de la aplicación.

- El **paquete constants**, contiene diferentes constantes que se usan para diferentes procedimientos en las vistas de usuario.

- El **paquete dialogs**, contiene diferentes diálogos que se muestran en la aplicación.

- El **paquete fragments**, contiene los diferentes fragments que se usan a lo largo del flujo de ejecución.

- El **paquete interfaces**, contiene diferentes interfaces que se implementan en ciertos módulos de la aplicación.

- El **paquete model**, contiene modelos de datos útiles para las vistas de usuario.

- El **paquete utils**, ofrece ciertas utilidades comunes para todos los módulos de las vistas.

Para organizar las vistas de la aplicación, se ha decidido que las vistas de usuario se implementen en 4 activities principales:

- **Login activity:** actividad destinada al inicio de sesión.

- **Register activity:** actividad destinada al registro de usuarios.

- **MenuListActivity:** actividad destinada a mostrar el menú de navegación principal.

- **MenuDetailActivity:** actividad destinada a mostrar en su interior el fragment correspondiente a la opción seleccionada en el menú principal.

Como se ha seguido el patrón master-detail, cuando estamos en una tableta, la actividad MenuListActivity, muestra el menú y a la vez el fragment seleccionado. En cambio, en dispositivos móviles, se muestra la activity MenuDetailActivity con el fragment correspondiente.

En cuanto a los fragments, se ha implementado un fragment por cada pantalla de navegación diferente de las existentes en el menú, permitiendo así tener la lógica de cada pantalla en un módulo concreto.

Para las distintas recycler view existentes, se han implementado distintos adapters, cada uno con las características y diseño necesario.

Como justificación a esta estructura, se ha pensado en un posible futuro crecimiento de la aplicación y un posible refactor de esta a otro tipo de comunicaciones como podría ser Firebase. Con el diseño actual, se podría cambiar el paquete rest por uno nuevo, haciendo que las modificaciones sobre las vistas fueran mínimas.

Dado este planteamiento, solo habría que modificar las llamadas a la API por las peticiones al nuevo paquete, pero se podría mantener toda la lógica de navegación.

Además, como existe la hipótesis de que cuando la aplicación empiece a ser utilizada saldrán errores, es mucho más fácil arreglar y aplicar cambios en módulos acotados que en un código que este todo mezclado.

En cuanto a las vistas, se recomienda no usar una activity para cada pantalla, sobretodo en contextos en que se comparten elementos y en los que existen elementos comunes entre varias vistas (por ejemplo, la barra de navegación superior).

Para lidiar con este posible overhead de memoria que puede provocar tener tantas activities, cada pantalla del menú se ha encapsulado en un fragment, dispuesto en una activity llamada MenuDetailActivity en el caso de dispositivos móviles y directamente en la pantalla MenuListActivity en tabletas.

Como hemos comentado anteriormente, esto también permite que un mismo fragment se pueda utilizar en dos sitios sin tener que duplicar el código, agilizando la implementación del patrón master-detail.

Para realizar una posible mejora de navegación de la aplicación en alguna de sus pantallas, bastaría con ir al fragment que la implementa y añadir la nueva funcionalidad, evitando que otras vistas se vean afectadas. En resumen, esto se traduce en una menor dependencia entre vistas y en una menor posibilidad de introducir errores colaterales.

Para la implementación del menú principal, inicialmente se pensó en añadir un botón por cada opción, pero se vio que, de cara a una futura escalabilidad, era mejor implementarlo como una recycler view a partir de una lista de texto. De esta forma, para añadir un nuevo ítem al menú, es suficiente con añadir el texto en la lista y automáticamente ya se muestra la nueva opción en el menú, evitando tocar el layout.

6.5 Arquitectura del código

En este apartado, se incluyen los diagramas de clases principales de la aplicación para poder visualizar la relación que existe entre los módulos diseñados.

Los módulos, se han pensado para que sean flexibles y modulares, evitando que existan dependencias complejas que puedan provocar errores y dificultad de mantenimiento.

Siguiendo las bases de los principios S.O.L.I.D, cada clase esta pensada para ser responsable de una solo objeto o pantalla. De esta manera, cada módulo, gestiona su propia lógica y evita posibles conflictos en otros modulos cuando se realiza un cambio.

Continuando con el mismo principio, el código esta diseñado para evitar que, si modificamos alguna interface, no se deba de implementar en sitios donde no se va a utilizar.

Por último, la capa rest se ha basado en el principio de inversión de dependencias, haciendo que las llamadas remotas sean transparentes para las vistas de usuario y facilitando un posible futuro cambio de servicio.

Diagramas del paquete REST:

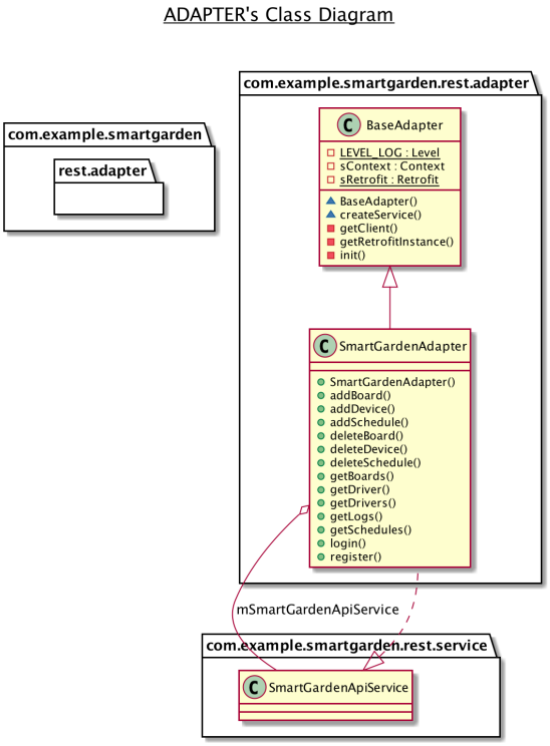


Figura 35: Diagrama de clases del paquete adapter

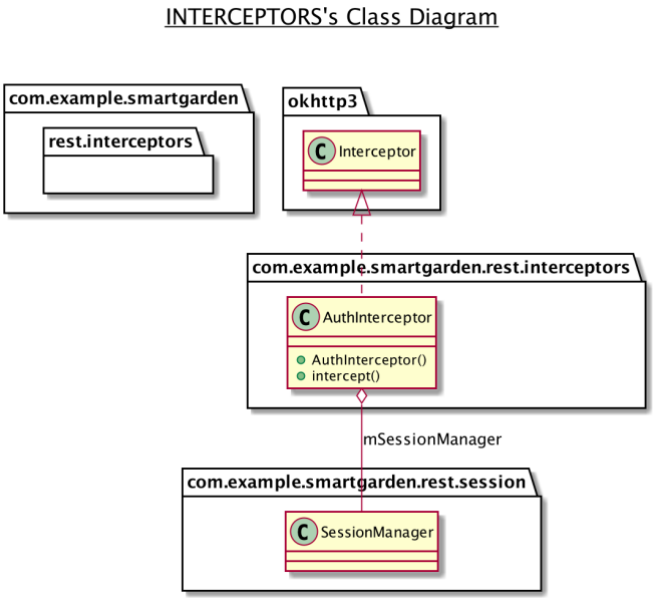


Figura 36: Diagrama de clases del paquete interceptors

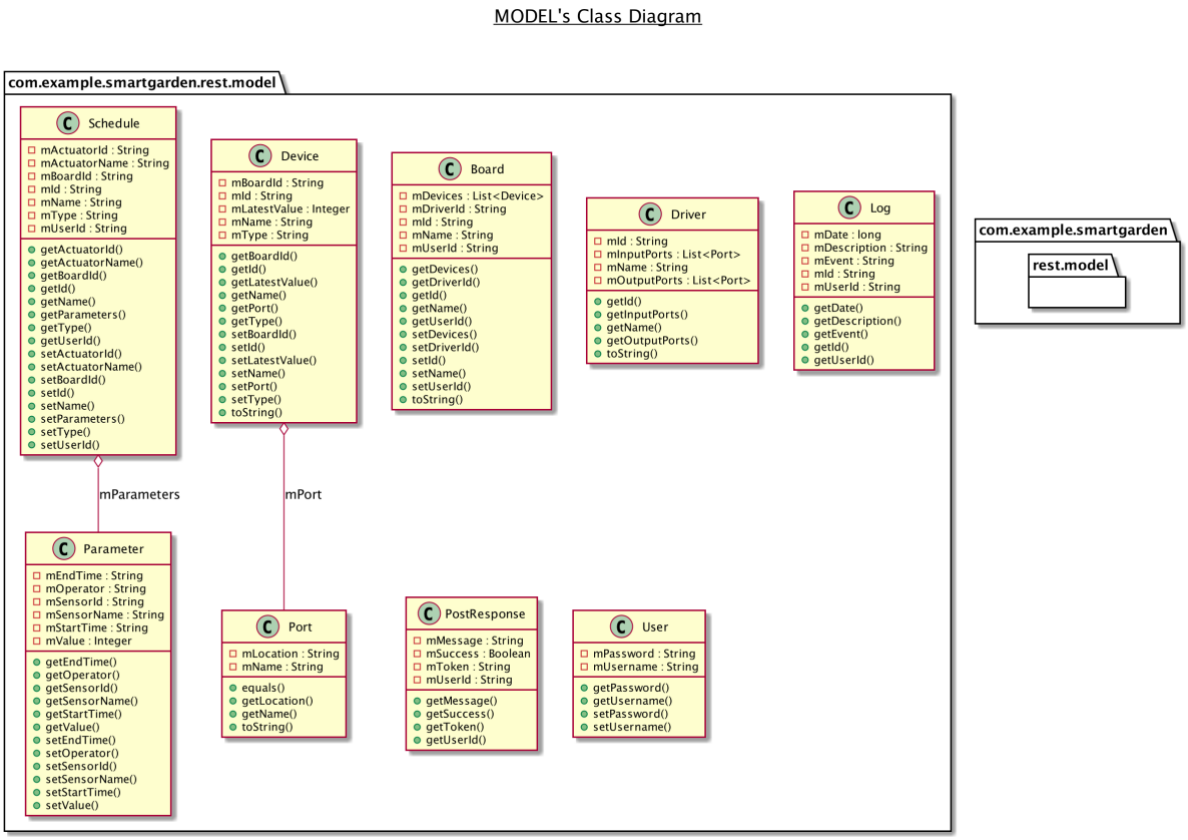


Figura 37: Diagrama de clases del paquete model

Diagramas del paquete UI:

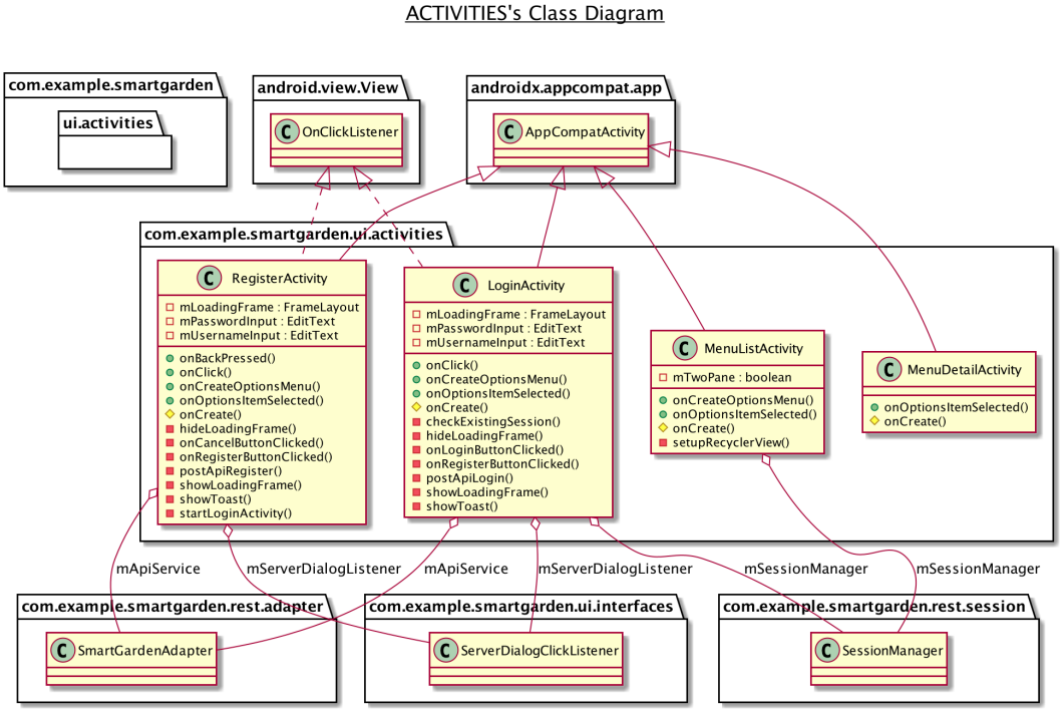


Figura 38: Diagrama de clases del paquete activities

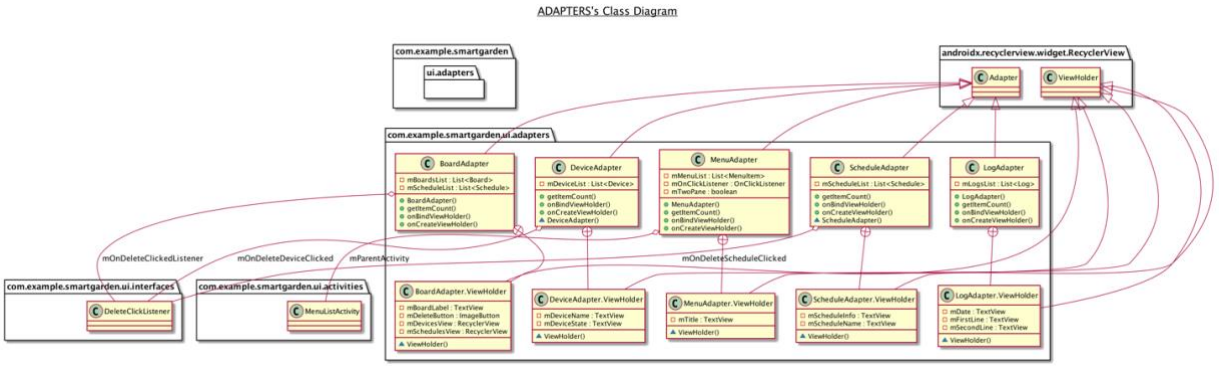


Figura 39: Diagrama de clases del paquete adapters

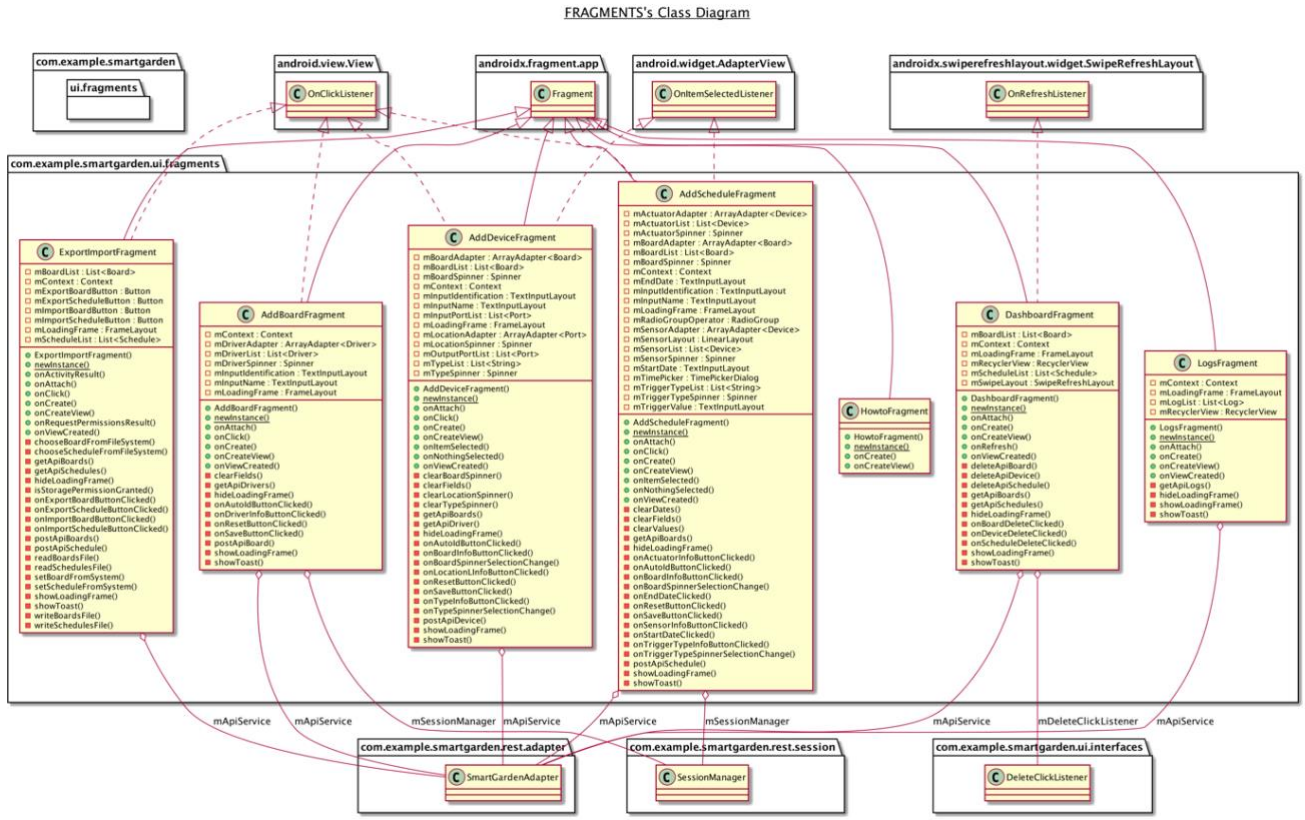


Figura 40: Diagrama de clases del paquete fragments

7. Validación funcional de la aplicación

7.1 Plan de validación

Para comprobar el correcto funcionamiento de la aplicación, se ha diseñado un plan de validación funcional, que se ha sido ejecutado por el desarrollador y por una persona externa al proyecto. Todos los test se han ejecutado en un dispositivo móvil i en una tableta comprobando así, que no existen errores de manera fiable.

En la Figura 41, se definen los diferentes casos de prueba.

Id	Caso de prueba	Funcionalidad
SG-CP1	Iniciar sesión un usuario registrado.	Inicio de sesión
SG-CP2	Iniciar sesión un usuario no registrado.	Inicio de sesión
SG-CP3	Iniciar sesión sin conexión al servidor.	Inicio de sesión
SG-CP4	Registrar un usuario que no exista.	Registro de usuario
SG-CP5	Registrar un usuario existente.	Registro de usuario
SG-CP6	Registrar un usuario sin conexión al servidor.	Registro de usuario
SG-CP7	Añadir una placa al sistema correctamente.	Añadir placa
SG-CP8	Añadir una placa al sistema con un identificador existente.	Añadir placa
SG-CP9	Añadir una placa sin conexión al servidor.	Añadir placa
SG-CP10	Añadir un dispositivo al sistema correctamente.	Añadir dispositivo

SG-CP11	Añadir un dispositivo al sistema con un identificador existente.	Añadir dispositivo
SG-CP12	Añadir un dispositivo al sistema sin conexión al servidor.	Añadir dispositivo
SG-CP13	Añadir una rutina por tiempo al sistema con fecha de inicio anterior a fecha de finalización.	Añadir rutina
SG-CP14	Añadir una rutina por evento al sistema con valor más grande que 0 y mas pequeño que 100.	Añadir rutina
SG-CP15	Añadir una rutina al sistema con fecha de inicio posterior a fecha de finalización.	Añadir rutina
SG-CP16	Añadir una rutina por evento al sistema con valor más pequeño que 0 o más grande que 100.	Añadir rutina
SG-CP17	Añadir una rutina con un identificador existe.	Añadir rutina
SG-CP18	Borrar una rutina con conexión al servidor.	Dashboard
SG-CP19	Borrar una rutina sin conexión al servidor.	Dashboard
SG-CP20	Borrar un dispositivo con conexión al servidor.	Dashboard
SG-CP21	Borrar un dispositivo sin conexión al servidor.	Dashboard
SG-CP22	Borrar una placa con conexión al servidor.	Dashboard
SG-CP23	Borrar una placa sin conexión al servidor.	Dashboard

SG-CP24	Exportar una configuración de placas con conexión al servidor.	Exportar / Importar
SG-CP25	Exportar una configuración de placas sin conexión al servidor.	Exportar / Importar
SG-CP26	Exportar una configuración de rutinas con conexión al servidor.	Exportar / Importar
SG-CP27	Exportar una configuración de rutinas sin conexión al servidor.	Exportar / Importar
SG-CP28	Importar una configuración de placas con conexión al servidor.	Exportar / Importar
SG-CP29	Importar una configuración de placas sin conexión al servidor.	Exportar / Importar
SG-CP30	Importar una configuración de rutinas con conexión al servidor.	Exportar / Importar
SG-CP31	Importar una configuración de rutinas sin conexión al servidor.	Exportar / Importar
SG-CP32	Importar una configuración de placas de un fichero erróneo.	Exportar / Importar
SG-CP33	Importar una configuración de rutinas de un fichero erróneo.	Exportar / Importar
SG-CP34	No dar permiso de acceso al almacenamiento externo.	Exportar/Importar
SG-CP35	Mantener la sesión iniciada al cerrar y abrir la aplicación.	General
SG-CP36	Cerrar sesión de usuario.	General

Figura 41: Tabla de definición de casos de prueba

7.2 Resultado de la validación

En la Figura 42, se especifican los resultados esperados de cada prueba y el resultado obtenido.

Id	Resultado esperado	Resultado
SG-CP1	Se visualiza el menú principal.	PASS
SG-CP2	Aparece un mensaje indicando que se ha podido iniciar sesión y se mantiene la pantalla de inicio de sesión.	PASS
SG-CP3	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea.	PASS
SG-CP4	Aparece un mensaje indicando que el registro ha sido exitoso y se visualiza la pantalla de inicio de sesión.	PASS
SG-CP5	Aparece un mensaje indicando que ya existe un usuario con ese nombre y se mantiene la pantalla de registro.	PASS
SG-CP6	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea.	PASS
SG-CP7	Aparece un mensaje indicando que la placa se ha añadido correctamente. En dashboard se visualiza la placa añadida.	PASS
SG-CP8	Aparece un mensaje indicando que el identificador ya existe. Se mantienen los datos introducidos.	PASS
SG-CP9	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea.	PASS
SG-CP10	Aparece un mensaje indicando que el dispositivo se ha añadido correctamente. En dashboard se visualiza el dispositivo añadido.	PASS
SG-CP11	Aparece un mensaje indicando que el identificador ya existe. Se mantienen los datos introducidos.	PASS

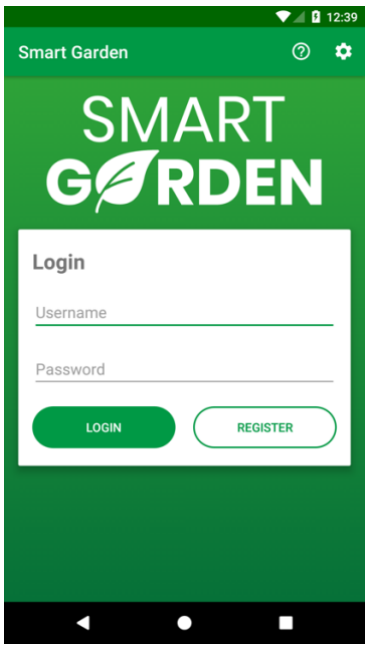
SG-CP12	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea.	PASS
SG-CP13	Aparece un mensaje indicando que la rutina se ha añadido correctamente. En dashboard se visualiza la rutina añadida.	PASS
SG-CP14	Aparece un mensaje indicando que la rutina se ha añadido correctamente. En dashboard se visualiza la rutina añadida.	PASS
SG-CP15	Aparece un mensaje indicando que la fecha de finalización debe ser posterior a la de inicio. Se mantienen los datos introducidos.	PASS
SG-CP16	Aparece un mensaje indicando que los valores deben estar por encima de 0 y tener como valor máximo 100. Se mantienen los datos introducidos.	PASS
SG-CP17	Aparece un mensaje indicando que el identificador ya existe. Se mantienen los datos introducidos.	PASS
SG-CP18	Aparece un mensaje indicando que se ha borrado y se han vuelto a obtener los datos correctamente.	PASS
SG-CP19	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea. No se realiza ninguna acción.	PASS
SG-CP20	Aparece un mensaje indicando que se ha borrado y se han vuelto a obtener los datos correctamente.	PASS
SG-CP21	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea. No se realiza ninguna acción.	PASS
SG-CP22	Aparece un mensaje indicando que se ha borrado y se han vuelto a obtener los datos correctamente.	PASS
SG-CP23	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea. No se realiza ninguna acción.	PASS
SG-CP24	Aparece un mensaje indicando que se ha exportado el fichero correctamente.	PASS

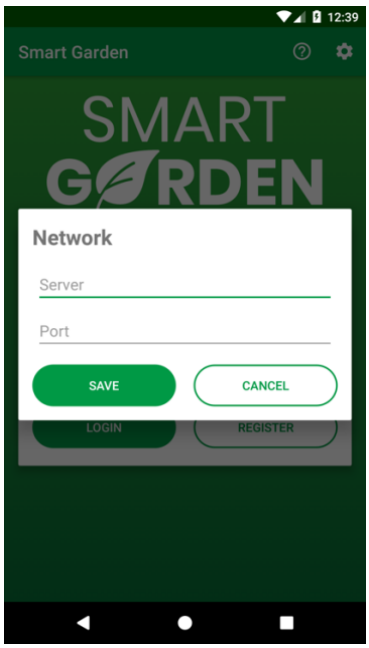
SG-CP25	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea. No se realiza ninguna acción.	PASS
SG-CP26	Aparece un mensaje indicando que se ha exportado el fichero correctamente.	PASS
SG-CP27	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea. No se realiza ninguna acción.	PASS
SG-CP28	Aparece un mensaje indicando que se ha importado el fichero correctamente.	PASS
SG-CP29	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea. No se realiza ninguna acción.	PASS
SG-CP30	Aparece un mensaje indicando que se ha importado el fichero correctamente.	PASS
SG-CP31	Aparece un mensaje indicando que la conexión con el servidor ha sido errónea. No se realiza ninguna acción.	PASS
SG-CP32	Aparece un mensaje indicando que el fichero contiene información de placa errónea. No se realiza ninguna acción.	PASS
SG-CP33	Aparece un mensaje indicando que el fichero contiene información de rutina errónea. No se realiza ninguna acción.	PASS
SG-CP34	Aparece un mensaje indicando que se requieren permisos. Se visualizan todos los botones deshabilitados.	PASS
SG-CP35	Se mantiene la sesión iniciada una vez se abre la aplicación por segunda vez.	PASS
SG-CP36	Se visualiza la pantalla de inicio de sesión.	PASS

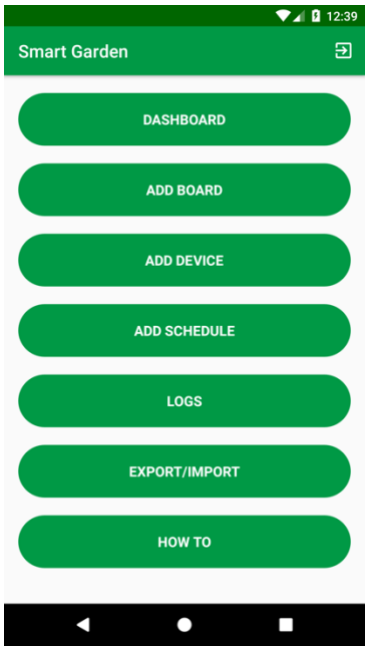
Figura 42: Tabla de resultados de los casos de prueba

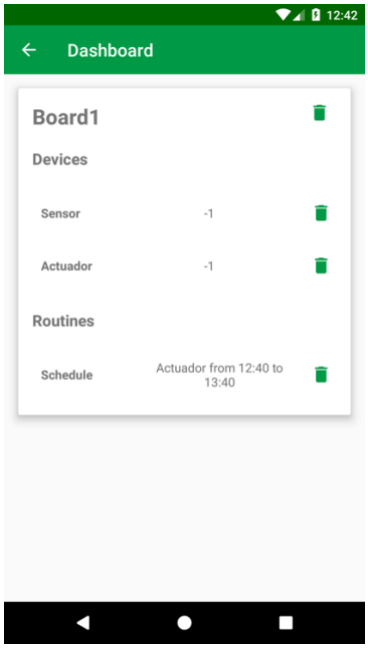
8. Diseño final de la aplicación

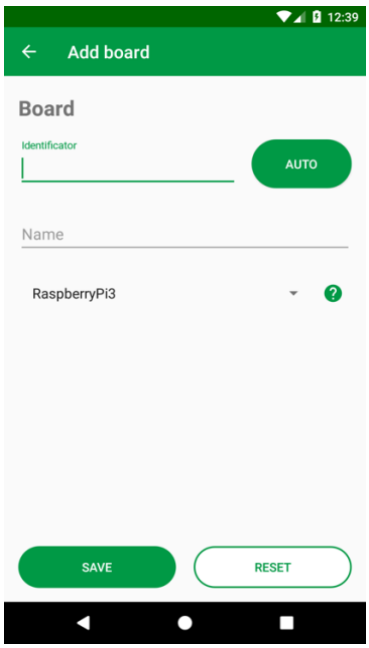
En este apartado, se puede visualizar como ha quedado la aplicación una vez finalizada su fase de diseño, implementación y validación.

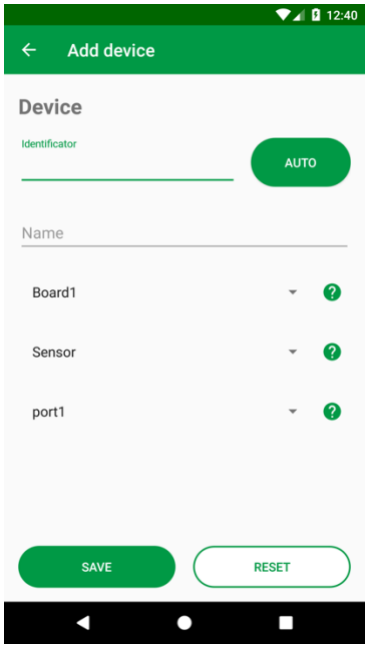












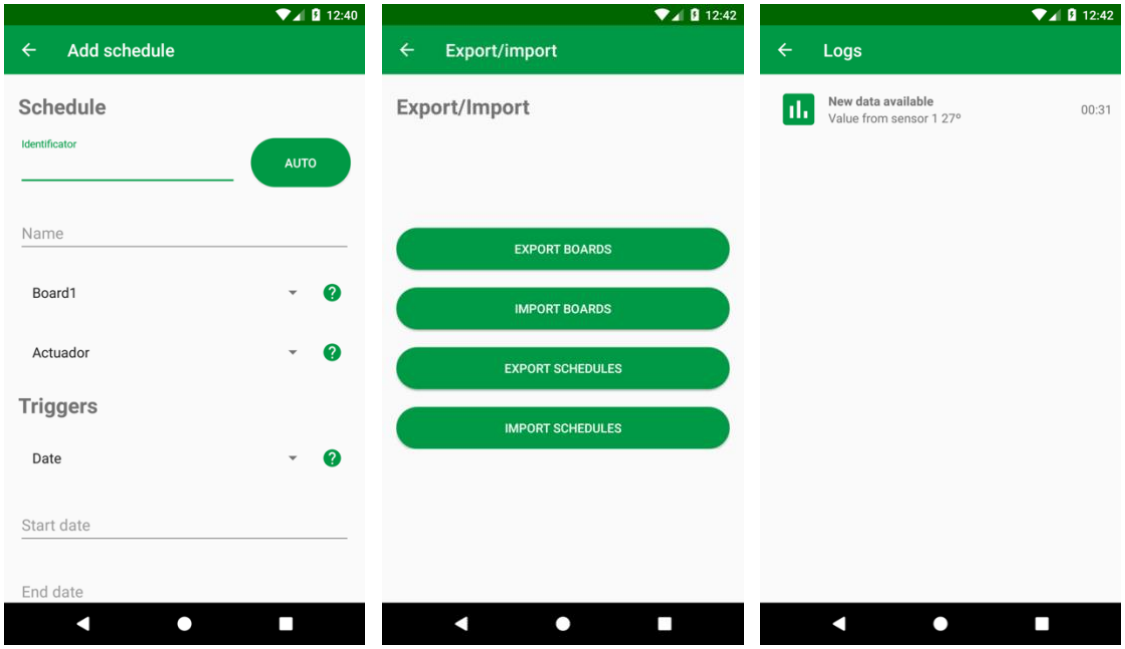


Figura 43: Capturas del diseño final de la aplicación

9. Conclusiones

9.1 Resultados

Después de finalizar el desarrollo del proyecto de final de máster, se ha concluido con que gran parte de los requisitos planteados inicialmente han sido satisfechos. Es cierto, que algunos requisitos no se han podido completar por falta de tiempo o se han completado parcialmente, pero en comparación con lo logrado, la valoración es positiva.

En cuanto al producto final obtenido, se han logrado implementar todas las funcionalidades que se planteaban al inicio del proyecto. Además, la aplicación final tiene una interfaz de usuario con detalles visuales muy cuidados.

Se ha adaptado a dispositivos móviles y tabletas la interfaz, tanto en posición vertical como horizontal. De este modo, el destinatario final es mucho más amplio que si partimos de una aplicación solo para móvil.

Los botones de las pantallas en las que se requieren permisos específicos o acciones no permitidas se han desactivado para guiar al usuario y hacer que la navegación sea mas sencilla. Además, en gran parte de los campos de las pantallas, se ha incluido un botón de información, que permite entender porque aparecen unos datos y no otros.

En cuanto a los detalles visuales, se ha diseñado un logotipo especial y se ha escogido una paleta de colores que se ha usado para todos los ítems de la aplicación.

Para aportar comodidad al usuario y evitar errores, los teclados que se muestran contienen únicamente los caracteres permitidos para el campo seleccionado. Además, para mejorar la introducción de fechas, se ha usado un selector especifico de Android. También, se ha acotado el numero de caracteres que se permiten en cada campo y se realiza una validación especifica para cada entrada de texto.

Para el desarrollo del código, este se ha estructurado pensando en futuras mejoras, haciendo que sea fácilmente escalable.

Se han separado los elementos de comunicación y vista para que, si en un futuro se quiere migrar a otro tipo de comunicaciones, esto sea transparente para las vistas de usuario. También, se ha separado el modelo de datos, haciendo que fácilmente se puedan introducir nuevos campos o elementos.

Por último, se ha elegido usar las librerías androidx y una SDK que ha permitido que la aplicación sea compatible con un mayor numero de dispositivos móviles.

9.2 Errores

Uno de los principales errores detectados ha sido querer cumplir demasiados objetivos en poco tiempo. Inicialmente, se hizo una estimación de tiempo y se creyó que sería posible abarcar más de lo posible.

El proyecto, constaba de 3 partes, la aplicación Android, el servidor web y una aplicación Python para las placas Raspberry Pi. Finalmente, solo se ha podido desarrollar la aplicación Android, el servidor (con algunos requisitos no cumplidos) y una aplicación Python de muestra, que solo sería para demostración del producto.

Otro error que creo que hubiera ayudado a mejorar el desarrollo del proyecto, fue la falta de un plan de contingencia ante imprevistos. Debido a la situación sanitaria vivida durante estos dos últimos meses, ha sido bastante complicado poder cumplir con las tareas programadas en las fechas planificadas. Por esto, se ha tenido que hacer un esfuerzo final mayor para conseguir finalizar con éxito el proyecto.

Para este proyecto se ha utilizado una estructura basada en Rest, debido a que era la más familiar para mi. Durante el transcurso del máster, se han trabajado alternativas como Firebase, pero mi nivel de conocimiento me pareció no ser suficiente para poder alcanzar las metas planteadas en el tiempo disponible.

9.3 Propuesta de mejora

En cuanto a mejoras a realizar en el proyecto, se plantean primeramente los diferentes requisitos funcionales y no funcionales no alcanzados:

- **SG-RF17** Recibir notificaciones cuando se inicia una rutina de riego.
- **SG-RF18** Recibir notificaciones cuando se finaliza una rutina de riego.
- **SG-RNF5** Las comunicaciones con la API se realizarán utilizando el protocolo HTTPS.

Creo que este último requisito, es muy importante, ya que los datos de los usuarios no deberían verse nunca comprometidos. Es imprescindible que antes de salir al mercado, la aplicación y el servidor usen la capa TLS.

Otras mejoras a nivel de desarrollo, sería la inclusión de pruebas unitarias aparte de las pruebas funcionales que ya se han realizado en esta fase del proyecto.

Como mejoras funcionales adicionales, se plantean los siguientes puntos:

- Los textos de la aplicación se han preparado para ser traducibles, faltaría contar con las traducciones y un apartado para poder seleccionar el idioma deseado.
- Añadir una pestaña de configuración de usuario que permitiera modificar la contraseña.
- Implementar un mecanismo de recuperación de contraseña en caso de olvidarla.
- Mejorar la seguridad de los JWT, haciendo que tengan fecha de expiración y obligando al usuario a renovar la sesión periódicamente.
- Mejorar la gestión de rutinas y no permitir que las horas de las rutinas o los valores sean incompatibles con otros previamente configurados.
- Mejorar la gestión de identificadores de las placas, los dispositivos y las rutinas, haciendo que sean únicos por usuario y no únicos por colección de la base de datos.
- Mejorar el formato de exportación de ficheros a JSON en vez de TXT y permitir solo la importación de ficheros JSON.

Como propuesta de futuro, creo que sería interesante plantear traducir la aplicación a Kotlin y migrarla a dispositivos iOS.

10. Glosario

Término	Definición
IOT	Internet of things
Industria 4.0	Industria caracterizada por el proceso de fabricación inteligente.
Raspberry Pi	Ordenador incrustado en forma de placa de bajo coste.
Arduino	Placa de desarrollo hardware usada para diseñar dispositivos digitales.
Gantt	Herramienta gráfica usada para representar el tiempo de realización de tareas.
API Rest	Representational State Transfer
JWT	JSON Web Token
GSON	Librería Java para convertir objetos a JSON.
UML	Unified Modeling Language
JSON	JavaScript Object Notation
HTTPS	HyperText Transfer Protocol Secure
Open source	Software con licencia de dominio público.
Android	Sistema operativo para móviles desarrollado por Google.
SDK	Software Development Kit
Fragment	En aplicaciones Android, porción de interfaz de usuario representada dentro de una Activity.
Activity	En aplicaciones Android, pantalla de la interfaz de usuario.
S.O.L.I.D	Conjunto de 5 directrices para facilitar la creación de código legible y mantenible.

11. Bibliografía

***Android Developers Documentation*.** (Abril de 2020).
Obtenido de https://developer.android.com/docs

***BodyParser NodeJS Documentation*.** (Abril de 2020).
Obtenido de https://www.npmjs.com/package/body-parser

***Express Documentation*.** (Abril de 2020).
Obtenido de https://expressjs.com/en/guide/routing.html

***GSON Documentation*.** (Mayo de 2020).
Obtenido de https://sites.google.com/site/gson/gson-user-guide

***JWT Documentation*.** (Abril de 2020).
Obtenido de https://jwt.io/introduction/

***JWT NodeJS Documentation*.** (2020 de Abril).
Obtenido de https://www.npmjs.com/package/jsonwebtoken

***MongoDB Documentation*.** (Abril de 2020).
Obtenido de https://docs.mongodb.com

***MongoDB NodeJS Driver Documentation*.** (Abril de 2020).
Obtenido de https://mongodb.github.io/node-mongodb-native/3.5/

***Morgan NodeJS Documentation*.** (Abril de 2020).
Obtenido de https://github.com/expressjs/morgan

***NodeJS Documentation*.** (Abril de 2020).
Obtenido de https://nodejs.org/es/docs/

***Requests Documentation*.** (Mayo de 2020).
Obtenido de https://requests.readthedocs.io/en/master/

***RestfulApi Documentation*.** (Abril de 2020).
Obtenido de https://restfulapi.net/http-status-codes/

***Retrofit2 Documentation*.** (Abril de 2020).
Obtenido de https://square.github.io/retrofit/

Wikipedia. (Mayo de 2020).
Obtenido de https://es.wikipedia.org/wiki/SOLID

12. Anexos

Junto con la memoria del proyecto, se entregarán los siguientes documentos:

- Manual de montaje del entorno
- Manual de usuario
- Código fuente del servidor
- Código fuente de la aplicación desarrollada
- Código fuente del script de Python para testeo del funcionamiento
- APK de la aplicación para móviles y tabletas
- Javadoc de la aplicación Android (carpeta docs dentro del proyecto)
- Presentación del proyecto

El código fuente del proyecto se puede descargar en las siguientes direcciones:

- **Servidor:**
  - https://github.com/xalepo4/express-server.git

- **Aplicación Android:**
  - https://github.com/xalepo4/android-hmi.git

- **Python Wrapper**
  - https://github.com/xalepo4/python-wrapper.git